pyOAL

January 27, 2025

Abstract

The class will create objects containing the basic information from the observation as read in the XML file created by (odfParamCreator).

1 Use

In order to use pyOAL simply import it in a Python session:

```
from pysas.pyOAL import pyOAL

obs = pyOAL.OBservation(path_to_XML_file)
```

A further explanation of the different functions and classes present in pyOAL can be found in the section below. For a more detailed description on how to run each function, use the help(...) command.

2 Description

pyOAL is a module containing several classes (Observation, Instrument, Exposure, Product, Task, Param) that allows the user to navigate through all the information in an observation by reading a XML file.

There are moree specific functions in the class that are discussed in the section ??.

2.1 Current classes

At the moment, these are the current functions and classes present in pyOAL:

- *Observation:* main class. Loads the data from the XML and saves them. Object from this class have the *_obs* functions that returns all the products from the observation.
- *Instrument:* Instrument class. Objects from this class contain exposures (*get_exposures*).

- *Exposure:* Exposure class. Objects from this class contain the basic information in their attributes. Products of each exposure can be obtained using *get_products*.
- *Product:* Products class.
- *Task:* Task class. Objects of this class contain the basic information regarding classes. Also contains *run_task*, which can be used alongside the list of parameters from the given task to directly run it with a Python interface.
- *Param:* Returns the raw values (as a list) of the column of the FITS file passed as argument.

2.2 Current general functions

By using the *super()* function in Python all children classes can access their parents methods. In addition there are some extra functions:

- *brancher:* returns all the lower branches of the XML starting from the reference branch given as an argument.
- *pareent_child_map:* (from the Observation class) returns a complete parent - child dictionary with the observation data.
- *create_xml:* creates an XML file with the same structure as the input file used to create Observation objects but with the updated data from the passed Observation.

2.3 Errors

Will raise the usual Python exceptions through the AstroPy/NumPy frame. Other errors will be notified to the user accordingly in each function.

3 Input Files

1. To create an Observation object, an XML created by (odfParamCreator).
2. All classes posses the get_() function in which a list containing all the objects from the class below that matches certain condition.

4 Output Files

1. Each utility has its own output files (or any). This is mentioned in the documentation for each function.

5 Comments

- Please report any bug found or any extra utilities that may seem useful for the purpose of SAS and/or (pyOAL).

References