PyStar

PyStar is a class library for data manipulation provided in the Self Defining Text Archival and Retrieval (STAR). The STAR syntax provides a way for simple, easy-to-comprehend, flexible and extensible data exchange (Hall, 1991, Hall, 1994). The syntax permits most types of data items, data structures and data cells. The STAR format is the basis for the Crystallographic Information File (CIF), which is widely used in crystallography for data archiving and exchange.

Folder	Description
PyStar	library
example	examples
docs	documentation

Main Features

- a simple data manipulation;
- a manipulation with comment text;

Installation of *PyStar*

PyStar is developed and tested using Python 3.7. They can be installed by:

```
python -m pip install ...  # as root (in Windows OS)
```

If you have downloaded a source tarball you can install PyStar by doing the following:

```
python setup.py develop # as root
```

And then just import a library in a python code:

```
import pystar
```

Usage

Reading a STAR File to an object:

```
import pystar
f_name = "example.cif"
star_object = pystar.read_star_file(f_name)
```

```
#or
star_object = pystar.read_file(f_name)
```

Access to ...

... item by item's name

```
#if data block is unique
item = star_object["_cell_length_a"]

#if data block is not unique
item = star_object["Fe3O4_cell_length_a"]
print(item.name, item.value, item.comment)
```

... items by items' prefix

```
items = star_object["_cell_length"]
print(items.names, items.values, items.comments)
```

... one-level loop by loop's prefix

```
loop = star_object["_atom_site"]
print(loop.names, loop.values, loop.comment_values, loop.comment_names)
```

Note: multi-level loops are not supported.

... data block by blockname

```
data_block = star_object["Fe3O4"]
print(data_block.name, data_block.items, data_block.loops, data_block.saves,
data_block.comment)
```

A data manipulation

Creation ...

```
item = pystar.Item(name="_jh", value="sf", comment="jhklj")
loop = pystar.Loop(names=("_jh_2", "_jh_zzz"), values=(("1", "2"), ("11", "22"),
("111", "222")))

data_block = pystal.DataBlock()
```

```
data_block.app_item(item)
data_block.app_loop(loop)
```

Correction ...

```
star_object["_cell_length_a"] = 8.3
#or
star_object["_cell_length_a"].value = 8.3

star_object["_cell_length_a"].comment = "comment line"
```

Saving ...

```
data_block.save_to_star_file("out.cif")
#or
data_block.save("out.cif")
```

The syntax of the STAR File

In the section shorts extracts with small changes are reproduced from "International Tables for Crystallography, Volume G, Definition and Exchange of Crystallographic Data, Edited by Sydney Hall and Brian McMahon, First edition, published for the International Union of Crystallography by Springer 2005". A more rigorous description of the STAR File syntax is given in (Hall, 1991 and Hall, 1994).

A STAR File is a sequential file containing lines of standard ASCII characters. A file may be divided into any number of discrete sets of unique data items. Sets may be in the form of data blocks, global blocks or save frames.

Item

A data item is a data value and its associated data name. Each data item stored in a STAR File is specified with this combination. A data name (or tag) is the identifier of a data value and is a sequence of non-white-space characters starting with an underscore character "_". A data value is a text string.

```
_publication_author_name  "Patrick O’Connor"
_publication_author_address
; School of CSSE
UWA
;
```

Loop

A looped list consists of the keyword `loop_` followed by

- a sequence of data names (possibly with nested `loop_` constructs);
- a sequence of loop packets, each containing data values which are identified in the same order as the data names.

A looped list specifies a table of data in which the data names represent the 'header descriptors' for columns of data and the packets represent the rows in the table. Looped lists may be nested to any level. Each loop level is initialized with the `loop_` keyword and is followed by the names of data items in this level. Data values that follow the nested data declarations must be in exact multiples of the number of data names. Each loop level must be terminated with a `stop_`, except the outermost (level 1) which is terminated by either a new data item or the privileged strings indicating a save frame, a data block, a global block or an end of file The attributes of data sets are as follows.

An example of a simple one-level loop structure is:

```
loop_
_atom_identity_number
_atom_type_symbol
1 C
2 C
3 O
```

Nested (multi-level) looped lists contain matching data packets and an additional `stop_` to terminate each level of data. Here is a simple example of a two-level nested list.

```
loop_
_atom_id_number
_atom_type_symbol
 loop_
 _atom_bond_id_1
 _atom_bond_id_2
 _atom_bond_order
  1 C    1  2 single      1 3 double stop_
  2 C    2  1 single stop_
  3 O    3  1 double stop_
```

Save frame

A save frame is a set of unique data items wholly contained within a data block. The frame starts with a `save_framecode` statement, where the framecode is a unique identifying code within the data block. Each frame is closed with a `save_` statement.

A save frame has the following attributes:

- A save frame may contain data items and loop structures but not other save frames.
- The scope of the data specified in a save frame is the save frame in which it is specified.
- Data values in a save frame are distinct from any identical items in the parent data block.

- A save frame may be referenced within the data block in which it is specified using a data item with a value of `$framecode` .
- A frame code must be unique within a data block.
- A save frame may not contain another save frame, but it may contain references to other save frames in the same data block using frame codes

```
data_example

save_phenyl
_object_class   molecular_fragment

loop_
_atom_identity_node
_atom_identity_symbol
 1 C 2 C 3 C 4 C 5 C 6 C
save_

loop_ _molecular_fragments $ethyl $phenyl $methyl
```

Data block

A data block is a set of data containing any number of unique items and save frames. A data block begins with a `data_` block code statement, where block code is a unique identifying name within a file. A data block is closed by another `data_` block code statement, a `global_` statement or an end of file.

A data block has the following attributes:

- A block code must be unique within the file containing the data block.
- Data blocks may not be referenced from within a file.
- The scope of data specified in a data block is the data block. The value of a data item is always associated with the data block in which it is specified.
- Data specifications in a data block are unique, except they may be repeated within a save frame. Data specifications in a save frame are independent of the parent data block specifications.
- If a data item is not specified in a given data block, the global value is assumed. If a global value is not specified, the value is unknown.

Global block

A global block is a set of data items which are implied to be present in all data blocks which follow in a file, unless specified explicitly within a data block. A global block starts with a `global_` keyword and is closed by a `data_` blockcode statement or an end of file.

A global block has the following attributes:

- The scope of global data is from the point of declaration to the end of file.
- A global block may contain data items, loop structures and save frames.
- Multiple global blocks are concatenated to form a single block in which the last item specification has precedence.

- A data item specified within a data block has precedence over a data item specified in a prior global block.

Data sets and scopes

A data set is the generic term for a unique set of data. A STAR File may contain three types of data sets: global blocks,data blocks and save frames. The attributes of data sets are as follows.

- A file may contain any number of data sets.
- The data names defined within a data set must be unique to that set. That is, all `data_` blockcode names must be unique within the file, all data names must be unique within a `global_` block, all data names and `save_framecode` s must be unique within a data block, and all data names must be unique within a save frame.
- The scope of data sets is hierarchical. Global blocks encompass all following data blocks; data blocks scope all contained save frames.
- The scope of a save frame is all data items contained within the frame.
- The scope of a data block is the boundaries of the data block,i.e. the end of the file or the start of the next data block, including any contained save frames. The same data item may be defined within a save frame and within the parent data block. All specifications of this item will be recognized when accessing the data block.
- The scope of a global block is the file, from the point of invocation to the end of file or the start of the next global block. It encompasses all contained global data items, data blocks and save frames. Globally specified data are active provided identical items are not specified in subsequent data sets.

Privileged constructs

The following constructs are privileged.

- Text strings starting with the character sequences `data_`, `loop_`, `global_`, `save_` or `stop_` are privileged words (keywords) and may not be used as values in text strings.
- A sharp character "#" (ASCII 35) is an explicit end-of-line signal provided it is not contained within a text string. Characters on the same line and following an active sharp character are considered as comment text.

Using `stop_` in looped lists

The `stop_` construction can be applied in the looped list of data names to terminate a loop of data values and to return the looped list to the next outer nesting level. The following, although not particularly intuitive, is a valid construction.

```
loop_
_atom_id_number
 loop_
 _atom_bond_id_1
 _atom_bond_id_2
 _atom_bond_order stop_
_atom_type_symbol
1    1 2 single     1 3 double   stop_  C
```

```
2    2 1 single  stop_  C
3    3 1 double  stop_  O
```

Collaboration

If you have any suggestions, bug reports or annoyances please report them to our issue tracker at site.

Copyright and License