mmcif$_{pdbx}$

Release 2.0.1

Apr 24, 2021

Contents

This is yet another PyPI package for http://mmcif.wwpdb.org/pdbx-mmcif-home-page.html. It emphasizes a simple and pure Python interface to basic mmCIF functionality.

The canonical mmCIF Python package can be found at https://github.com/rcsb/py-mmcif. It is full-featured and includes C/C++ code to accelerate I/O functions.

This package provides the module `pdbx`. More information about the `pdbx` module can be found in the *API reference* section.

CHAPTER 1

Origin of this software

All of the code in this repository is based on http://mmcif.wwpdb.org/. Specifically, this code is directly derived from http://mmcif.wwpdb.org/docs/sw-examples/python/src/pdbx.tar.gz linked from http://mmcif.wwpdb.org/docs/sw-examples/python/html/.

See http://mmcif.wwpdb.org/docs/sw-examples/python/html/ for more information about this package, including examples.

CHAPTER 2

Versions

Versions 0.* maintain API compatibility with the original code. Subsequent versions break that compatibility, primarily by renaming methods in compliance with PEP8.

CHAPTER 3

Installation

This python package can be installed via setuptools, `pip install .`, or via PyPI.

CHAPTER 4

Testing

The software can be tested with pytest by running:

```
python -m pytest
```

from the top-level directory.

CHAPTER 5

More information

More information about this software can be found in the documentation. Guidelines for community behavior are provided in the code of conduct and information on contributing to the software is provided in our contribution guide.

CHAPTER 6

Indices and tables

- genindex
- modindex

CHAPTER 7

Contents

7.1 API reference

Note: The API is still changing. We use semantic versioning and our *Change log* to document changes between versions.

7.1.1 Basic functions

PDBx/mmCIF Python dictionary resources.

All of the code in this repository is original based on http://mmcif.wwpdb.org/. Specifically, this code is directly derived from the pdbx code linked from PDBx Python Parser Examples and Tutorial.

See PDBx Python Parser Examples and Tutorial for more information about this package, including examples.

pdbx.**dump**(*datacontainers*, *fobj*)
Write a list of objects to a CIF file.

Parameters

- **datacontainers** (*list*) – a list of *DataContainer* objects # noqa E501
- **fobj** (*file*) – a file object ready for writing

pdbx.**dumps**(*datacontainers*) → str
Serialize a list of objects to a CIF-formatted string.

Parameters **datacontainers** (*list*) – list of *DataContainer* objects # noqa E501

Returns CIF-formatted string

pdbx.**load**(*fobj*) → list
Parse a CIF file.

Parameters **fobj** (*file*) – file object ready for reading

Returns a list of *DataContainer* objects

pdbx.**loads**(*text*) → list
Parse a CIF string.

Parameters **s** (*str*) – string with CIF data

Returns a list of *DataContainer* objects

7.1.2 Input-output classes

reader

PDBx/mmCIF dictionary and data file parser.

Note: Acknowledgements:

The tokenizer used in this module is modeled after the clever parser design used in the PyMMLIB package.

PyMMLib Development Group:

Authors: Ethan Merritt: merritt@u.washington.edu, Jay Painter: jay.painter@gmail.com

See: http://pymmlib.sourceforge.net/

---

class pdbx.reader.**PdbxReader**(*input_file*)
PDBx reader for data files and dictionaries.

read(*container_list*)
Appends to the input list of definition and data containers.

Parameters **container_list** (*list*) – list of *ContainerBase* containers to append to.

writer

Classes for writing data and dictionary containers in PDBx/mmCIF format.

class pdbx.writer.**PdbxWriter**(*output_file=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Write PDBx data files or dictionaries. Use the input container or container list.

set_row_partition(*num_rows*)
Maximum number of rows checked for value length and format.

Parameters **num_rows** (*int*) – maximum number of rows

write(*container_list*)
Write out a list of containers.

Parameters **container_list** (*list*) – list of *ContainerBase* objects to write.

write_container(*container*)
Write out information for an individual container.

Parameters **container** (*ContainerBase*) – container to write

7.1.3 Data structure classes

containers

A collection of container classes supporting the PDBx/mmCIF storage model.

A base container class is defined which supports common features of data and definition containers. PDBx data files are organized in sections called data blocks which are mapped to data containers. PDBx dictionaries contain definition sections and data sections which are mapped to definition and data containers respectively.

Data in both PDBx data files and dictionaries are organized in data categories. In the PDBx syntax individual items or data identified by labels of the form '_categoryName.attribute_name'. The terms category and attribute in PDBx jargon are analogous table and column in relational data model, or class and attribute in an object oriented data model.

The DataCategory class provides base storage container for instance data and definition meta data.

class pdbx.containers.**CifName**
Class of utilities for CIF-style data names.

static attribute_part(*name*) → str
Get the attribute part of the name.

Parameters **name** (*str*) – name

Returns attribute part of name

static category_part(*name*) → str
Get the category part of the name.

Parameters **name** (*str*) – name

Returns category part of name

class pdbx.containers.**ContainerBase**(*name*)
Container base class for data and definition objects.

append(*obj*)
Add the input object to the current object catalog. An existing object of the same name will be overwritten.

Parameters **obj** (*DataCategory*) – input object to catalog

exists(*name*) → bool
Determine if object name exists in object catalog.

Parameters **name** (*str*) – object name

Returns whether object exists in object catalog

get_object(*name*)
Get object from object catalog.

Parameters **name** (*str*) – object name

Returns object or None

Return type *DataCategory*

get_object_name_list() → list
Get list of object names.

Returns list of *DataCategory* objects

get_type() → str
Get container type.

Returns container type

name
Get container name.

Returns container name

print_it (*fobj=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *type_='brief'*)
Dump information about container to specified file object.

Parameters

- **fobj** (*file*) – file object for writing
- **type** (*str*) – type of summary ("brief" makes it short)

remove (*current_name*) → bool
Remove object by name.

Parameters **current_name** (*str*) – name of object to remove

Returns True on success or False otherwise.

rename (*current_name*, *new_name*) → bool
Change the name of an object in place.

Parameters

- **current_name** (*str*) – old name for object
- **new_name** (*str*) – new name for object

Returns indicator of whether renaming was successful

replace (*obj*)
Replace an existing object with the input object.

Parameters **obj** (*DataCategory*) – input object to catalog

set_name (*name*)
Set container name.

Parameters **name** (*str*) – container name

set_type (*type_*)
Set container type.

Parameters **type** (*str*) – container type

class pdbx.containers.**DataCategory** (*name*, *attribute_name_list=None*, *row_list=None*)
Methods for creating, accessing, formatting PDBx cif data categories.

append (*row*)
Add row to container.

Parameters **row** (*list*) – row to add

append_attribute (*attribute_name*)
Add attribute to container.

Parameters **attribute_name** (*str*) – name of attribute to add

append_attribute_extend_rows (*attribute_name*)
Append attribute and extend rows.

Parameters **attribute_name** (*str*) – name of attribute to add

attribute_count
Get number of attributes.

attribute_list
Get list of attributes.

attribute_list_with_order
Get list of attributes in order.

current_attribute
Get current attribute.

current_row_index
Get current row index.

dump_it(*file_=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Dump contents of container.

Parameters **file** (*file*) – file object ready for writing

get_attribute_index(*attribute_name*) → int
Get index of given attribute.

Parameters **attribute_name** (*str*) – name of attribute

Returns index of attribute

Raises **IndexError** – if attribute not found

get_format_type_list(*steps=1*) → str
Get a formatted type list.

Parameters **steps** (*int*) – step size for iterating through rows

Returns formatted type list

get_format_type_list_x
Alternate version of format type list.

get_full_row(*index*) → list
Return a full row based on the length of the the attribute list.

Parameters **index** (*int*) – index of row to retrieve

Returns row

get_max_attribute_list_length(*steps=1*) → int
Get maximum length of attribute value list.

Parameters **steps** (*int*) – step size for iterating through rows

Returns attribute value list max length

get_row(*index*) → list
Get specified row.

Parameters **index** (*int*) – row index

Returns specified row or empty array if row not found.

get_value(*attribute_name=None*, *row_index=None*)
Get value for specified attribute and row.

Parameters

- **attribute_name** (*str*) – attribute name
- **row_index** (*int*) – row index

Returns attribute value

Raises **IndexError** – if attribute not found

get_value_formatted(*attribute_name=None*, *row_index=None*) → str
Get formatted version of value.

Parameters

- **attribute_name** (*str*) – attribute name
- **row_index** (*int*) – row index

Returns formatted value

get_value_formatted_by_index(*attribute_index*, *row_index*) → str
Get value formatted by index.

Parameters

- **attribute_name** (*str*) – attribute name
- **row_index** (*int*) – row index

Returns formatted value

has_attribute(*attribute_name*) → bool
Indicate whether container has attribute.

invoke_attribute_method(*attribute_name*, *method*)
Invoke method of current attribute.

Parameters

- **attribute_name** (*str*) – attribute name
- **method** (*str*) – name of attribute method

invoke_category_method(*method*)
Invoke method of current category.

Parameters **method** (*str*) – name of method

item_name_list
List of attribute names as fully qualified item names.

max_attribute_list_length
Get maximum attribute list length.

name
Get container name.

print_it(*file_=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)
Print container information.

Parameters **file** (*file*) – file object ready for writing

remove_row(*index*) → bool
Remove specified row.

Parameters **index** (*int*) – index of row to remove

Returns True if successful, False otherwise

rename_attribute(*current_attribute_name*, *new_attribute_name*) → bool
Change the name of an attribute in place.

Parameters

- **current_attribute_name** (*str*) – current attribute name

- **new_attribute_name** (*str*) – new attribute name

Returns flag indicating renaming success

replace_substring(*old_value*, *new_value*, *attribute_name*) → bool
Replace substring of value of given attribute.

Parameters

- **old_value** – old attribute value
- **new_value** – new attribute value
- **attribute_name** (*str*) – name of attribute to replace

Returns Boolean flag indicating success.

replace_value(*old_value*, *new_value*, *attribute_name*) → int
Replace the value of the specified attribute.

Parameters

- **old_value** – old attribute value
- **new_value** – new attribute value
- **attribute_name** (*str*) – name of attribute to replace

Returns number of replacements

row_count
Get number of rows.

row_list
Get list of rows.

set_value(*value*, *attribute_name=None*, *row_index=None*)
Set value of attribute.

Parameters

- **value** – value of attribute to set
- **attribute_name** (*str*) – name of attribute
- **row_index** (*int*) – index of row

class pdbx.containers.**DataCategoryBase**(*name*, *attribute_name_list=None*, *row_list=None*)
Base object definition for a data category.

get() → tuple
Get name, attribute name list, and row list.

Returns tuple of (name, attribute name list, and row list)

set_attribute_name_list(*attribute_name_list*)
Set attribute name list.

Parameters **attribute_name_list** (*list*) – list of attribute names

set_name(*name*)
Set name.

Parameters **name** (*str*) – object name to set

set_row_list(*row_list*)
Set row list.

Parameters **row_list** (*list*) – list of rows

class pdbx.containers.**DataContainer**(*name*)
Container class for DataCategory objects.

get_global() → bool
Return global flag.

invoke_data_block_method(*method*)
Invoke a method for the given data block.

Parameters **method** (*str*) – name of method

set_global()
Set global flag to True.

class pdbx.containers.**DefinitionContainer**(*name*)
Container for definitions.

is_attribute() → bool
Determine if container contains item objects.

Returns indicator of whether item objects are in container

is_category() → bool
Determine if container contains category objects.

Returns indicator of whether category objects are in container

print_it(*file_=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *type_='brief'*)
Print information about container to file object.

Parameters

- **file** (*file*) – file object for writing
- **type** (*str*) – type of summary ("brief" makes it short)

7.1.4 Error-handling classes

errors

Error classes for PDBx/mmCIF.

exception pdbx.errors.**PdbxError**
Class for general errors.

exception pdbx.errors.**PdbxSyntaxError**(*line_number*, *text*)
Class for syntax errors.

7.2 Change log

7.2.1 v2.0.1

Additions

- Added testing for Python 3.9

Changes

- Removed versioneer
- Add more detail to documentation. (#34)

Fixes

- Fixed versioning numbers in code and documentation (#48)

7.2.2 v2.0.0 (15-Dec-2020)

- Added a change log. (#42)
- Fix PyPI release.

7.2.3 v1.1.2 (01-Aug-2020)

Additions

- Added versioneer versioning support. (#39, #40)
- Added Sphinx API documentation and set up readthedocs.io site. (#8, #31, #35, #36)
- Added de-linting to continuous integration pipeline with psf/black. (#37)

Changes

- Flattened namespace. *This should have triggered a 1.2.0 release but the versioning wasn't updated correctly.* (#32)

7.2.4 v1.1.1 (11-Jul-2020)

Changes

- Minor update of version number in `setup.py`.

7.2.5 v1.1.0 (11-Jul-2020)

Additions

- Includes new tests. (#30)

Changes

- Implements new CIF I/O functions: load, loads, dump, dumps. (#28, #29)
- Improved consistency in reading and writing special characters. (#21, #27)

7.2.6 v1.0.0 (07-Jul-2020)

Changes

- Significant changes to API, including:
  - PEP8-complaint class and function naming
  - Improved *pdbx.reader* use of `StopIteration`. (#22, #23)
  - Simplification of module structure. (#19, #24, #26)
  - Removal of redundant tests and conversion of testing to pylint.
- General code de-linting. (#1, #6, #14)
- Updates to documentation. (#25)

Fixes

- Fix typo in continuous integration pipeline. (#17)

7.2.7 v0.0.1 (05-Jul-2020)

Initial release.

Python Module Index

p

Index

H

I

L

M

N

P

R

S

W