PyPDB

API Documentation

June 16, 2010

Contents

1 Module PyPDB10

A PDB class to parse a PDB file

Written (2001-2003) by P. Tuffery, INSERM, France Contributions by R. Gautier, J. Maupetit, J. Herisson, F. Briand Version: 10.0 (2010 June)

The idea is to have an easy management of PDB files

Ex:

x = PDB("1tim") y = PDB("1timA")

# 1st residue of x

x[0]

# Chain A of x

x["A"]

#1st atom of the 2nd residue of x

x[0][1]

Classes' Pattern:

PDBLine (a line of a PDB file) atmLine (a line ATOM/HETATM of a PDB file) atmList (a list of lines ATOM/HETATM of a PDB file (ex: a list of atoms lines)) residu (a residue of a file (atomic information)) PDB (a PDB)

PDBList (function, not a class for the moment) manages a collection of PDBs.

protein : Class to manage a protein type PDB (it is not finish at present (ongoing adjustments), but it is partially functional)

1.1 Functions

resType(*aName*)

Parameters

`aName`: a string of the sequence encoded using a 3 letters code

Return Value

the position of the string

Author: P. Tuffery

aa3Type(*aName*)

Parameters

`aName`: a string of the sequence encoded using a 3 letters code

Return Value

the position of the string

Author: P. Tuffery

aa1Type(*aName*)

Parameters

`aName`: a string of the sequence encoded using a 1 letter code

Return Value

the position of the string

Author: P. Tuffery

SEQRESToAA1(*seqres*, *verbose*=`0`)

PDB SEQRESToAA1

Parameters

`seqres`: a sequence encoded using a 3 letters code (separated by blanks)

Return Value

the sequence encoded using a 1 letter code (in a string)

aln2mask(*s1*, *s2*)

This will convert an alignement into a selection mask

Parameters

`s1`: a string

`s2`: a string

Note:

- s1 and s2 must be 2 strings of identical lengths.
- gaps (indels) must be represented by '-'

xyzOXT(*ca*, *cp*, *o*, *verbose*=`0`)

xyzOXT return terminal oxygen position

Parameters

`ca`: atmLine of the Carbon Alpha

`cp`: atmLine of the backbone oxygene (prime)

`o`: atmLine of the backbone oxygene

Return Value

tuple (x,y,z) with OXT coordinates

Author: F.Briand

PDBBiologicalUnit(*PDBid*=`None`, *verbose*=`0`)

PDBBiologicaUnit:

Parameters

`PDBid`: none, or a PDBid

Return Value

the PDB entry biological unit at PQS server (EBI: http://pqs.ebi.ac.uk/pqs-doc/macmol/2lzm.mmol)

PDBEntries(*what*=`None`, *isauthor*=`'no'`, *verbose*=`0`)

PDBEntries

Parameters

`what:` a word searched on PDB.org

`isauthor:` none, or an author

Return Value

a list of entries matching a word on http://www.pdb.org/pdb/navbarsearch.do?newSearch=yes&isAuthorSearch=no&radioset=All &inputQuickSearch=calpain&image.x=0&image.y=0&image=Search

CSASite2Escan(*PDBid*=`None`, *patterns*=`'strict'`, *purge*=`0`, *verbose*=`0`)

CSASite2Escan extract Catalytic Site atoms for a PDB. It gets the information directly at CSA.

Parameters

`PDBid:` a PDB file

`purge:` does it keep only compatible sites? (No: 0 by default)

`patterns:` one of "strict", "medium", "light"

(type="strict": exact atom name match (by default); "medium": atom name match using atom class compatible pattern!; "light" : atom name match using light maks (atomic type))

Return Value

a list of atoms involved.

Note: Will not take into account psiblast sites.

purgeSites(*sites*, *hetCheck*=`0`, *verbose*=`0`)

purgeSites only keep compatible sites

Parameters

`sites:` a list of sites

`hetCheck:` if 1, also check heteros

Return Value

the list of compatible sites.

identicalp(*site1*, *site2*, *hetCheck*=`0`, *verbose*=`0`)

Check if site1 is compatible with site2 on the basis of residue names, residue number and which part (sidechain, backbone)

Parameters

`hetCheck:` if 1, also check heteros

Return Value

the compatibility of the both sites (0: no Matches, 1: Matches)

samep(*site1*, *site2*, *hetCheck*=`0`, *verbose*=`0`)

Check if site1 is compatible with site2 on the basis of residue names, residue number and which part (sidechain, backbone)

Parameters

`hetCheck`: if 1, also check heteros

Return Value

the compatibility of the both sites (0: no Matches, 1: Matches)

EscanCASSites(*PDBid*=`None`, *patterns*=`'strict'`, *purge*=`0`, *verbose*=`0`)

extract Catalytic Site atoms for a PDB. recurse to validated catalytic sites (follow referer if required). Then gets the information directly at CSA.

Parameters

`PDBid`: a PDB file

`purge`: does it keep only compatible sites? (No: 0 by default)

`patterns`: one of "strict", "medium", "light"

(type="strict": exact atom name match; "medium": atom name match using atom class compatible pattern; "light" : atom name match using light maks (atomic type))

Return Value

a list of atoms involved.

makeHName(*rName*, *index*, *norm*)

makeHName() return the hydrogen name for the H in position "index" of the entry "rName" of globals dictionnary relativ to the specified "norm"

Parameters

`rName`: residue name

`index`: index of the hydrogen in the "rName" line of "normHNames dictionnaries

`norm`: formating norm, "IUPAC" or "PDB"

Return Value

a 4 characters string corresponding to the hydrogen name formated with "norm" rules

Author: F.Briand

PDBList(*input*=`None`, *hetSkip*=`0`, *altCare*=`0`, *OXTCare*=`0`, *verbose*=`0`)

This is to organize the iterative treatment of PDB instances.

Parameters

`input:` a list (of lines), or a file (list of lines).

(type=each line can specify: a local file, it may contain a multi PDB separated with HEADER / END lines aPDB Id compatible with the PDB class an url)

`altCare:` does the file contains the alternate atoms (No:0 by default)

`OXTCare:` does the file contains the information of each line (No:0 by default)

`hetSkip:` does the file skip the hetero atoms (No:0 by default)

Return Value

a list of PDB instances.

Note: The list length is the number of lines of the input

fileInput(*input*=`None`, *hetSkip*=`0`, *altCare*=`0`, *OXTCare*=`0`, *verbose*=`0`)

fileInput: to read a multiPDB file from disk.

Parameters

`input:` the file to read

`altCare:` does the file contains the alternate atoms (Yes:1 by default)

`OXTCare:` does the file contains the information of each line (No:0 by default)

`hetSkip:` does the file skip the hetero atoms (No:0 by default)

Return Value

a parsed PDB instance of the inputs contents

parseInput(*inputs*, *Id*=`None`, *hetSkip*=`0`, *altCare*=`0`, *OXTCare*=`0`, *verbose*=`0`)

parseInput: We have a list of PDBlines. We parse them and return a list of PDBs.

Parameters

`inputs:` the PDB instance to parse

`Id:` the Id of the protein

`altCare:` does the file contains the alternate atoms (Yes:1 by default)

`OXTCare:` does the file contains the information of each line (No:0 by default)

`hetSkip:` does the file skip the hetero atoms (No:0 by default)

Return Value

a parsed PDB instance of the inputs contents

outPDBList(*pdbList*, *outName*=`''`, *initMode*=`'w'`, *altCare*=`0`, *altLbl*=`''`, *OXTCare*=`0`, *hetSkip*=`0`, *fmode*=`'w'`, *header*=`1`, *ter*=`1`, *end*=`1`, *info*=`0`, *verbose*=`0`)

outPDBList

Parameters

`pdbList:` a list of PDB

`initMode:` mode of creating file, ”w” write by default

`outName:` the name of file where the PDB list will be written

`fmode:` mode of opening file, ”w” write by default

Return Value

none, it writes the PDB list in a file

PDBSumHeaders(*what*)

PDBSumHeaders

Parameters

`what:` the word that you want to search on EBI website

Return Value

the PDB ids corresponding to your research on what

PDBListFromPDBSum(*what*, *hetSkip*=`0`, *altCare*=`0`, *OXTCare*=`0`, *verbose*=`0`)

PDBListFromPDBSum

Parameters

`what:` the word that you want to search on EBI website

`altCare:` does the file contains the alternate atoms (No:0 by default)

`OXTCare:` does the file contains the information of each line (No:0 by default)

`hetSkip:` does the file skip the hetero atoms (No:0 by default)

Return Value

a list of all the PDB ids corresponding to your research on what

subPDB(*pdb*, *seedSeq*)

Given a PDB instance, return a part corresponding to the seed sequence

Parameters

`pdb:` a PPDB instance

`seedSeq:` the sequence to fetch (a string)

Return Value

a PDB instance corresponding to the seeSeq or None if the seedSeq is not found

Author: P. Tuffery

1.2 Variables

Name	Description
AA1	**Value:** `'ACDEFGHIKLMNPQRSTVWY'`

continued on next page

Name	Description
AA3	**Value:** `['ALA', 'CYS', 'ASP', 'GLU', 'PHE', 'GLY', 'HIS', 'ILE', ...`
AA1seq	**Value:** `'ACDEFGHIKLMNPQRSTVWYXXXSXWMCXWYMDPECXXYAMMSCMA'`
AA3STRICT	**Value:** `['ALA', 'CYS', 'ASP', 'GLU', 'PHE', 'GLY', 'HIS', 'ILE', ...`
AA3new	**Value:** `['PAQ', 'AGM', 'PR3', 'DOH', 'CCS', 'GSC', 'GHG', 'OAS', ...`
dico_AA	**Value:** `{'143': 'C', '1LU': 'L', '2AS': 'D', '2LU': 'L', '2MR': '...`
RNA3	**Value:** `['U']`
DNA3	**Value:** `['A', 'T', 'G', 'C']`
SOLV	**Value:** `['HOH', 'H2O', 'WAT', 'DOD']`
BBATMS	**Value:** `['N', 'CA', 'C', 'O', 'OXT']`
SCATMS	**Value:** `['-', 'N', 'CA', 'C', 'O', 'OXT']`
NCHIS	**Value:** `[0, 1, 2, 3, 2, 0, 2, 2, 4, 2, 3, 2, 0, 3, 5, 1, 1, 1, 2, 2]`
CHIATMS	**Value:** `[[], [['N', 'CA', 'CB', 'SG']], [['N', 'CA', 'CB', 'CG'],...`
AASC	**Value:** `[['CB'], ['CB', 'SG'], ['CB', 'CG', 'OD1', 'OD2'], ['CB',...`
AABB	**Value:** `['N', 'CA', 'C', 'O']`
GBINPATH	**Value:** `'/home/tintin/tuffery/bin/'`
GHMMPATH	**Value:** `'/data/HMM/models/HMM1/'`

continued on next page

Name	Description
normHNames	

```
# OLD IUPACHNames
HNames = {
        "ALA" : ["HB1","HB2","HB3"],
        "CYS" : ["HB1","HB2","HG"],
        "ASP" : ["HB1","HB2"],
        "GLU" : ["HB1","HB2","HG1","HG2"],
        "PHE" : ["HB1","HB2","HD1","HE1","HZ","HE2","HD2"],
        "GLY" : ["HA2"],
        "HIS" : ["HB1","HB2","HD2","HE1","HD1"],
        "ILE" : ["HB","HG11","HG12","HD11","HD12","HD13","HG21","HG22
        "LYS" : ["HB1","HB2","HG1","HG2","HD1","HD2","HE1","HE2","HZ1
        "LEU" : ["HB1","HB2","HG","HD11","HD12","HD13","HD21","HD22",
        "MET" : ["HB1","HB2","HG1","HG2","HE1","HE2","HE3"],
        "ASN" : ["HB1","HB2","HD21","HD22"],
        "PRO" : ["HB1","HB2","HG1","HG2","HD1","HD2"],
        "GLN" : ["HB1","HB2","HG1","HG2","HE21","HE22"],
        "ARG" : ["NH1","NH2","HB1","HB2","HG1","HG2","HD1","HD2","HE"
        "SER" : ["HB1","HB2","HG"],
        "THR" : ["HB","HG1","HG21","HG22","HG23"],
        "VAL" : ["HB","HG11","HG12","HG13","HG21","HG22","HG23"],
        "TRP" : ["HB1","HB2","HD1","HE1","HZ2","HH2","HZ3","HE3"],
        "TYR" : ["HB1","HB2","HD1","HE1","HE2","HD2","HH"],
        "BCK" : ["HA","HN","HN1","HN2","HN3"]
        }

# OLD PDBHNames
PDBHNames = {
        "ALA" : ["1HB","2HB","3HB"],
        "CYS" : ["1HB","2HB"," HG"],
        "ASP" : ["1HB","2HB"],
        "GLU" : ["1HB","2HB","1HG","2HG"],
        "PHE" : ["1HB","2HB"," HD1"," HE1"," HZ"," HE2"," HD2"],
        "GLY" : ["2HA"],
        "HIS" : ["1HB","2HB"," HD2"," HE1"," HD1"],
        "ILE" : [" HB","1HG1","2HG1","1HD1","2HD1","3HD1","1HG2","2HG
        "LYS" : ["1HB","2HB","1HG","2HG","1HD","2HD","1HE","2HE","1HZ
        "LEU" : ["1HB","2HB"," HG","1HD1","2HD1","3HD1","1HD2","2HD2"
        "MET" : ["1HB","2HB","1HG","2HG","1HE","2HE","3HE"],
        "ASN" : ["1HB","2HB","1HD2","2HD2"],
        "PRO" : ["1HB","2HB","1HG","2HG","1HD","2HD"],
        "GLN" : ["1HB","2HB","1HG","2HG","1HE2","2HE2"],
        "ARG" : ["1HB","2HB","1HG","2HG","1HD","2HD"," HE","1HH1","2H
        "SER" : ["1HB","2HB"," HG"],
        "THR" : [" HB"," HG1","1HG2","2HG2","3HG2"],
        "VAL" : [" HB","1HG1","2HG1","3HG1","1HG2","2HG2","3HG2"],
        "TRP" : ["1HB","2HB"," HD1"," HE1"," HZ2"," HH2"," HZ3"," HE3
        "TYR" : ["1HB","2HB"," HD1"," HE1"," HE2"," HD2"," HH"],
        "BCK" : [" HA"," H","1H","2H","3H"]
        }
```

Value: `{'lIUPAC': ({'ALA': ['HB1', 'HB2', 'HB3'], 'ARG': ['HB2',...`

continued on next page

Name	Description
BODY_WIDTH	**Value:** `0`
False	**Value:** `0`
GDFLTCATHDIR	**Value:** `'/data/banks/CATH/current/pdb'`
GDFLTPDBDIR	**Value:** `'/data/pdb/data/structures/'`
GDFLTSCOPDIR	**Value:** `'/data/banks/Astral/current/'`
HAADBIN	**Value:** `'/home/briand/Documents/PDBpy/HAAD/HAAD'`
LINKS_EACH_PARAGRAPH	**Value:** `0`
REDUCEBIN	**Value:** `'/home/briand/Documents/PDBpy/Reduce/reduce.3.14.080821.l...`
True	**Value:** `1`
UNICODE_SNOB	**Value:** `0`
__package__	**Value:** `None`
k	**Value:** `'icirc'`
r_unescape	**Value:** `re.compile(r'&(#?[xX]?(?:[0-9a-fA-F]+|\w{1,8}));')`
unifiable	**Value:** `{'aacute': 'a', 'acirc': 'a', 'aelig': 'ae', 'agrave': 'a...`
unifiable_n	**Value:** `{160: ' ', 169: '(C)', 183: '*', 224: 'a', 225: 'a', 226:...`

1.3 Class PDBLine

Known Subclasses: PyPDB10.PDB, PyPDB10.atmLine

PDBLine : basic management (mostly to access line type (ATOM, REMARK, etc) for one text line of a PDB datafile.

1.3.1 Methods

__init__(*self*, *aLine*=`''`)

PDBLine.__init__

Parameters

`aLine`: the text sets on the line

Return Value

none

__getslice__(*PDBLine*, *ffrom*=`0`, *tto*=`None`)

Parameters

`ffrom`: the first residu considered

`tto`: the last residu of the residues (excluded, as in python)

Return Value

a string, a slice of residues

__repr__(*self*)

PDBLine.__repr__

Return Value

print the PDBLine

__getitem__(*PDB*, *aPos*)

Parameters

`aPos`: the position of one residue (PDB[aPos])

Return Value

one residue

__len__(*self*)

PDBLine.__len__

Return Value

the length of the PDBLine

flat(*self*)

PDB.flat

Return Value

the string of the current line

header(*self*)

PDBLine header

Return Value

the six first chars of the line (string)

Note: could be one of:

- HEADER
- REMARK
- ATOM
- CONECT
- etc

1.4 Class atmLine

PyPDB10.PDBLine ─┐
PyPDB10.atmLine

Known Subclasses: PyPDB10.atmList

class atmLine This models one PDB ATOM / HETATOM line and its accessors

1.4.1 Methods

__init__(*self*, *aLine*=`''`)

atmLine.__init__

Parameters

`aLine:` the text sets on the line

(type=could be atmLine, PDBLine or a string)

Return Value

none

Overrides: PyPDB10.PDBLine.__init__

header(*self*, *hdr*=`''`)

PDB line HEADER

Parameters

`hdr:` none, or a new header

(type=must be string)

Return Value

ATOM or HETATM if the parameter hdr is not defined, otherwise it sets HETATM as header

Overrides: PyPDB10.PDBLine.header

atmNum(*self*, *anum*=`''`)

PDB line atom number

Parameters

`anum:` none, or an atom number

(type=anum may be int or string)

Return Value

the atom number (string) if there is no parameter, otherwise it sets anum as atom number

atmName(*self*, *aname*=`''`)

PDB line atom name

Parameters

`aname:` none, or an atom name

(type=aname must be string of size 4 at max, must begin by a blank if necessary !)

Return Value

the atom name (string) if there is no parameter, otherwise it sets aname as atom number

atmType(*self*, *atype=*`''`)

PDB line atom type

Parameters

`aname:` none, or an atom type

`atype:` *(type=atype must be string of size 2 at max, right aligned !)*

Return Value

the atom type (string) if there is no parameter, otherwise it sets atype as atom type

atmBVal(*self*, *abval=*`''`)

PDB line atom B value

Parameters

`abfact:` none, or an atom B value

`anum:` *(type=abval may be float or string)*

Return Value

the atom B value (string) if there is no parameter, otherwise it sets abfact as atom B value

alt(*self*, *acode=*`''`)

PDB line alternate code

Parameters

`acode:` none, or an alternate code

(type=acode must be one character)

Return Value

the alternate code (1 char string) if there is no parameter, otherwise it sets acode as alternate code

resName(*self*, *rName=*`''`)

PDB line residue name

Parameters

`rName:` none, or a residue name

(type=rName must be string (3 chars))

Return Value

the residue name (string) if there is no parameter, otherwise it sets residue name

chnLbl(*self*, *lbl*=`''`)

PDB line chain label

Parameters

`lbl`: none, or an atom chain label

(type=lbl must be 1 character)

Return Value

the atom chain label (1 character string) if there is no parameter, otherwise it sets the atom chain label

resNum(*self*, *rnum*=`''`)

PDB line residue number

Parameters

`rnum`: none, or a residue number rnum

(type=rnum may be string or int)

Return Value

the residue member (1 character string) if there is no parameter, otherwise it sets the atom residue number

resType(*self*, *verbose*=`0`)

PDB liste resType

Return Value

the type of the residues of the list

Note: it could be:

- AMINO-ACID
- RNA
- DNA
- SOLVENT
- HETERO

icode(*self*, *thecode*=`''`)

PDB line code number

Parameters

`thecode`: none, or a residue code

(type=the code must be 1 character)

Return Value

the residue code (1 character string) if there is no parameter, otherwise it sets the residue code

xyz(*self*)

PDB line xyz

Return Value

the atom's coordinated in 3D (x,y,z)

Note: x, y, z returned are float

crds(*self*)

PDB line crds

Return Value

a string containing the atom's coordinated in 3D (x,y,z), or 0.,0.,0. if the crds are not found

Note: x, y and z are contained in one string

setcrds(*self*, *x*, *y*, *z*)

PDB line set crds

Parameters

`x, y, z:` the atom's coordinated in 3D

Return Value

None

fpt(*self*, *aQ*=`''`)

PDB line occupancy

Parameters

`aQ:` none, or a new occupancy for the atom

(type=must be a float)

Return Value

the occupancy read on the PDB line or the new occupancy set by the user

q(*self*, *aQ*=`''`)

PDB line occupancy

Parameters

`aQ:` none, or a new occupancy for the atom

(type=must be a float)

Return Value

the occupancy read on the PDB line or the new occupancy set by the user

occ(*self*, *aOcc*=`''`)

PDB line occupancy

Parameters

`aOcc`: none, or a new occupancy for the atom

(type=must be a float)

Return Value

the occupancy read on the PDB line or the new occupancy set by the user

r(*self*, *aR*=`''`)

PDB line B_iso_or_equiv

Parameters

`aR`: none, or a new B_iso_or_equiv for the atom

(type=must be a float)

Return Value

the B_iso_or_equiv read on the PDB line or the new B_iso_or_equiv set by the user

tfac(*self*, *tFac*=`''`)

PBD line tfac

Parameters

`tFac`: none, or a new temperature factor for the atom

(type=must be a float)

Return Value

the temperature factor read on the PDB line or the new temperature factor set by the user

segId(*self*)

PBD line segId

Return Value

the segment Id

ele(*self*)

PDB line ele

Return Value

the element symbol

chrg(*self*)

PDB line type symbol

Return Value

the type symbol

Inherited from PyPDB10.PDBLine(Section 1.3)

__getitem__(), __getslice__(), __len__(), __repr__(), flat()

1.5 Class atmList

PyPDB10.PDBLine
PyPDB10.atmLine
PyPDB10.atmList

Known Subclasses: PyPDB10.residue

class atmList This models a list of PDB ATOM / HETATM lines (atmLine)

1.5.1 Methods

__init__(*self*, *data*=`''`, *chId*=`''`, *hetSkip*=`0`, *verbose*=`0`)

atmList.__init__ determine the type of data to initialize

Parameters

`data`: an instance

Return Value

none

Overrides: PyPDB10.PDBLine.__init__

__len__(*self*)

atmList.__len__

Return Value

the length of the atmList

Overrides: PyPDB10.PDBLine.__len__

__add__(*self*, *new*)

atmList.__add__

Parameters

`new`: the list added to the first one

Return Value

the two lists concatenated

__getslice__(*atmList*, *ffrom*=`0`, *tto*=`None`)

Parameters

`ffrom:` the first residu considered

`tto:` the last residu of the residues (excluded, as in python)

Return Value

a sub atmList, a slice of residues

Overrides: PyPDB10.PDBLine.__getslice__

__getitem__(*atmList*, *aPos*)

Parameters

`aPos:` the number of one atom in an atmList (PDB[residue][aPos])

Return Value

one atom line

Overrides: PyPDB10.PDBLine.__getitem__

__delitem__(*self*, *aPos*)

atmList.__delitem__

Parameters

`aPos:` the position of the atom to delete in the atmlist

Return Value

none

__setitem__(*self*, *aPos*, *new*)

atmList.__setitem__

Parameters

`aPos:` the position of the atom to insert

`new:` the new atom to insert in the position aPos

Return Value

none

__repr__(*self*, *altLbl*=`''`, *OXTSkip*=`0`, *HSkip*=`0`)

atmList.__repr__

Parameters

`OXTSkip:` does it skip OXT? (Yes:1 ; No:0)

`HSkip:` does it skip hydrogen? (Yes:1 ; No:0)

`altLbl:` alternate atom label

Return Value

show atomic information of the atmList in a string

Overrides: PyPDB10.PDBLine.__repr__

flat(*self*, *altLbl*=`''`, *OXTSkip*=`0`, *PDBMac*=`0`, *keepH*=`1`)

PDB list flat

Parameters

`altCare:` does it take care about alternate atom? (Yes:1 ; No:0)

`PDBMac:`

`altLbl:` alternate atom label

`OXTSkip:` does it skip OXT? (Yes:1 ; No:0)

`keepH:` does it keep the hydrogen atoms?

Return Value

a list of lines

Overrides: PyPDB10.PDBLine.flat

insert(*self*, *aPos*, *new*)

PDB list insert

Parameters

`aPos:` the position of new

`new:` the list inserted

Return Value

the list after insertion of new

crds(*self*, *ffrom*=`0`, *tto*=`-1`)

PDB List coordinates (crds)

Parameters

`ffrom`: the first atom of the range that we need coordinates

`tto`: the last one, -1 by default, in this case it will be equal to the last atom of the list

Return Value

a list of all coordinates of the range (concatenated)

Overrides: PyPDB10.atmLine.crds

xyz(*self*, *ffrom*=`0`, *tto*=`-1`)

PDB List xyz

Parameters

`ffrom`: the first atom of the range that we need coordinates

`tto`: the last one, -1 by default: in this case it will be equal to the last atom of the list

Return Value

a list of lists of coordinates x, y, z of the range

Overrides: PyPDB10.atmLine.xyz

BC(*self*, *ffrom*=`0`, *tto*=`-1`)

PDB List BC give the center of geometry of a collection of atoms

Parameters

`ffrom`: the first atom of the range that we need the center of geometry

`tto`: the last one, -1 by default: in this case it will be equal to the last atom of the list

Return Value

a list of coordinates x, y, z of the center of geometry

radius(*self*, *ffrom*=`0`, *tto*=`-1`)

PDB List radius

Parameters

`ffrom`: the first atom of the range that we need the center of geometry

`tto`: the last one, -1 by default: in this case it will be equal to the last atom of the list

Return Value

a list of coordinates x, y, z of the center of geometry, the maximum one found

oneChis(*self*)

PDB List oneChis

Return Value

the dihedral of the side chain

chis(*self*)

PDB List chis

Return Value

the dihedral of the side chain of the residues

outChis(*self*)

PDB List outChis

Return Value

none, print the dihedral of the side chain of the residues

atmPos(*self*, *aName*)

PDB List atmPos

Parameters

`aName`: name of the atom searched

Return Value

aPos the position of the first atom named ”aName”

Npos(*self*)

PDB List Npos

Return Value

aPos the position of the first atom N

CApos(*self*)

PDB List CApos

Return Value

aPos the position of the firstatom CA

Cpos(*self*)

PDB List Cpos

Return Value

aPos the position of the first atom C

Opos(*self*)

PDB List Opos

Return Value

aPos the position of the first atom O

out(*self*)

PDB List out

Return Value

none

Note: just ”pass”

resName(*self*)

PDB list residue name

Parameters

`rName:` none, or a residue name

Return Value

the residue name (string) if there is no parameter, otherwise it sets residue name

Overrides: PyPDB10.atmLine.resName

theAtm(*self*, *atmName*=`''`)

PDB list theAtm

Parameters

`atmName:` the name of the atom searched to know its line

Return Value

the line of atmName

isPDB(*self*)

PDB list isPDB:

Return Value

1

---

write(*self*, *outName*=`''`, *label*=`''`, *hetSkip*=`0`, *verbose*=`0`)

PDB List write PDB or PDB chain(s) to file

Parameters

`outName`: the name of the file written, if no name sets, it will be write on the standard out

`label`:

Return Value

none

Note: *for example:*

from PDB import *

x = protein("/home/raid5/PDB/pdb1acc.ent.gz",hetSkip=1)

x.frg(0).write()

---

oneHMMGeo(*self*, *aCA*)

PDB List oneHMMGeo

Parameters

`aCA`: an atom

Return Value

the seven descriptors of a fragment of one letter of the structural alphabet

Inherited from PyPDB10.atmLine(Section 1.4)

alt(), atmBVal(), atmName(), atmNum(), atmType(), chnLbl(), chrg(), ele(), fpt(), header(), icode(), occ(), q(), r(), resNum(), resType(), segId(), setcrds(), tfac()

1.6 Class residue

PyPDB10.PDBLine
PyPDB10.atmLine
PyPDB10.atmList
PyPDB10.residue

Known Subclasses: PyPDB10.PDB

class residue This models a list of PDB ATOM / HETATOM lines with semantic significance

1.6.1 Methods

__init__(*self*, *data*=`''`, *verbose*=`0`)

residue.__init__ determine the type of data to initialize the residue

Parameters

`data:` an instance

Return Value

none

Overrides: PyPDB10.PDBLine.__init__

__len__(*self*)

residue.__len__

Return Value

the length of the atmList

Overrides: PyPDB10.PDBLine.__len__

__repr__(*self*, *altCare*=`0`, *altLbl*=`''`, *OXTCare*=`0`, *HSkip*=`0`)

residue.__repr__

Parameters

`altCare`: does it take care about alternate atom? (Yes:1 ; No:0)

`OXTCare`: does it take care about OXT? (Yes:1 ; No:0)

`HSkip`: does it skip hydrogen? (Yes:1 ; No:0)

`altLbl`: alternate atom label

Return Value

show atomic information of the atmList in a string

Overrides: PyPDB10.PDBLine.__repr__

__getslice__(*residue*, *ffrom*=`0`, *tto*=`None`)

Parameters

`ffrom`: the first residu considered

`tto`: the last residu of the residues (excluded, as in python)

Return Value

a sub atmList, a slice of atoms

Overrides: PyPDB10.PDBLine.__getslice__

__getitem__(*residue*, *aPos*)

Parameters

`aPos`: the position of one residue (PDB[aPos]) or CHAINS (PDB["CHAINS"])

Return Value

one residue or PDB instance of chains matching CHAINS (e.g. "AB")

Overrides: PyPDB10.PDBLine.__getitem__

flat(*self*, *altCare*=`0`, *altLbl*=`''`, *OXTCare*=`0`, *PDBMac*=`0`, *keepH*=`1`)

residue.flat

Parameters

`altCare`: does it take care about alternate atom? (Yes:1 ; No:0)

`altLbl`: alternate atom label

`PDBMac`:

`OXTCare`: does it take care about OXT? (Yes:1 ; No:0)

Return Value

a string of the list of lines

Overrides: PyPDB10.PDBLine.flat

rName(*self*, *name*=`''`, *verbose*=`0`)

residue.rName give residue name

Parameters

`name`: none, or a residue name

(type=must be string (3 chars))

Return Value

the residue name (string) if there is no parameter, otherwise it sets residues name

rNum(*self*, *aNum*=`''`, *verbose*=`0`)

residue.rNum give residue number

Parameters

`aNum`: none, or a residue number aNum

(type=aNum may be string or int)

Return Value

the residue number (1 character string) if there is no parameter, otherwise it sets the atoms residue number

riCode(*self*, *icode*=`''`, *verbose*=`0`)

residue.riCode give code number

Parameters

`icode:` none, or a residue code

(type=the code must be 1 character)

Return Value

the residue code (1 character string) if there is no parameter, otherwise it sets the residues code

rType(*self*, *verbose*=`0`)

residue.rType give the type of residue

Return Value

the type of the residues of the list

Note: it could be:

- AMINO-ACID
- RNA
- DNA
- SOLVENT
- HETERO

chnLbl(*self*, *lbl*=`''`, *verbose*=`0`)

residue.chnLbl give or set chain label

Parameters

`lbl:` none, or an atom chain label

(type=lbl must be 1 character)

Return Value

the atom chain label (1 character string) if there is no parameter, otherwise it sets the atoms chain label

Overrides: PyPDB10.atmLine.chnLbl

atmPos(*self*, *aName*)

residue.atmPos give atmPos

Parameters

`aName:` name of the atom searched

Return Value

aPos the position of the first atom named "aName"

Overrides: PyPDB10.atmList.atmPos

hasAltAtms(*self*, *verbose*=`0`)

residue.hasAltAtms

Return Value

Does the file has BBaltAtm or SCAltAtm? (Yes/No for each)

altLbls(*self*, *verbose*=`0`)

residue.atlLbls

Return Value

all the alternate atoms of the atom list

select(*self*, *awhat*=`['']`)

residue.select

Parameters

`awhat:` what atoms

Return Value

a selection of atoms, and atmList

delete(*self*, *awhat*=`None`)

residue.delete

Parameters

`awhat:` none, or a list of atom names

Return Value

none

Note: This will remove atoms from the residue based on their names

BBAtmMiss(*residue*)

Return Value

the positions of all BB atms missing in the atm list

findAtm(*self*, *atmName*=`'CA'`, *chId*=`None`, *rName*=`None`, *rNum*=`None`, *icode*=`None`, *verbose*=`0`)

residue.findAtm identify an atom

Parameters

`atmName`: the atom name

`chId`: the chain Id

`rName`: the residu name

`rNum`: the residu number

`icode`: the line code number

Return Value

the atom instance

setBBOrder(*self*, *verbose*=`0`)

residue.setBBOrder set backbone's atoms of a residue in the "right" order

Return Value

none

Author: F.Briand

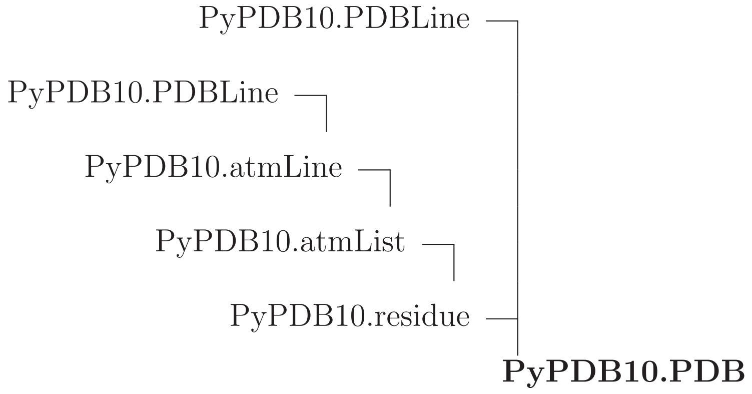
Inherited from PyPDB10.atmList(Section 1.5)

BC(), CApos(), Cpos(), Npos(), Opos(), __add__(), __delitem__(), __setitem__(), chis(), crds(), insert(), isPDB(), oneChis(), oneHMMGeo(), out(), outChis(), radius(), resName(), theAtm(), write(), xyz()

Inherited from PyPDB10.atmLine(Section 1.4)

alt(), atmBVal(), atmName(), atmNum(), atmType(), chrg(), ele(), fpt(), header(), icode(), occ(), q(), r(), resNum(), resType(), segId(), setcrds(), tfac()

1.7 Class PDB



Known Subclasses: PyPDB10.protein

PDB class Manage a PDB

1.7.1 Methods

__init__(*self*, *fname*=`''`, *chId*=`''`, *model*=`1`, *hetSkip*=`0`, *altCare*=`0`, *OXTCare*=`0`, *PDBMac*=`0`, *keepH*=`1`, *id*=`None`, *isXML*=`0`, *verbose*=`0`)

PDB.__init__ determine the type of fname to initialize

Parameters

`fname:`	the file name
`chId:`	a chain Id
`model:`	the number of the model you want to set as working model (number 1 by default)
`hetSkip:`	does the file skip the hetero atoms (No:0 by default)
`altCare:`	does it take care about alternate atom? (Yes:1 ; No:0)
`OXTCare:`	does it take care about OXT? (Yes:1 ; No:0)
`PDBMac:`	
`keepH:`	does it keep the hydrogen atoms?
`id:`	
`isXML:`	
`altLbl:`	alternate atom label

Return Value

none

Overrides: PyPDB10.atmLine.__init__

__getslice__(*PDB*, *ffrom*=`0`, *tto*=`None`)

Parameters

`ffrom:` the first residu considered

`tto:` the last residu of the residues (excluded, as in python)

Return Value

a PDB instance, a slice of residues

Overrides: PyPDB10.atmList.__getslice__

__getitem__(*PDB*, *aPos*)

Parameters

`aPos:` the position of one residue (PDB[aPos]) or CHAINS (PDB["CHAINS"])

Return Value

one residue or PDB instance of chains matching CHAINS (e.g. "AB")

Overrides: PyPDB10.atmList.__getitem__

__len__(*PDB*)

PDBLine.__len__

Return Value

number of residues of the PDB instance

Overrides: PyPDB10.atmList.__len__

__add__(*PDB*)

atmList.__add__

Parameters

`new:` a PDB to concatenate to a first one

Return Value

the 2 PDBs concatenated *in this example, it will return z:*

x = PDB()

new = PDB()

z = x + new

Overrides: PyPDB10.atmList.__add__

__delitem__(*self*, *aPos*)

del x[i]

Parameters

`aPos`: position of the residu

Return Value

none

Overrides: PyPDB10.atmList.__delitem__

Note: preserves anything in comments: modify atms then performs resTab() MIGHT BE BUGGY (P. Tuffery, 2007)

__repr__(*self*, *altCare*=`0`, *altLbl*=`''`, *OXTCare*=`0`, *HSkip*=`0`)

PDB.__repr__

Parameters

`altCare`: does it take care about alternate atoms? (Yes:1 ; No:0)

`OXTCare`: does it take care about OXT? (Yes:1 ; No:0)

`HSkip`: does it skip hydrogen? (Yes:1 ; No:0)

`altLbl`: alternate atom label

Return Value

show atomic information of PDB (print PDB ATOM lines)

Overrides: PyPDB10.atmList.__repr__

flat(*self*, *altCare*=`0`, *altLbl*=`''`, *OXTCare*=`0`, *PDBMac*=`0`, *keepH*=`1`)

PDB list flat

Parameters

`altCare`: does it take care about alternate atom? (Yes:1 ; No:0)

`altLbl`: alternate atom label

`OXTCare`: does it take care about OXT? (Yes:1 ; No:0)

Return Value

an atmList

Overrides: PyPDB10.atmList.flat

out(*self*, *outName*=`''`, *chainId*=`''`, *altCare*=`0`, *altLbl*=`''`, *OXTCare*=`0`, *hetSkip*=`0`, *fmode*=`'w'`, *header*=`1`, *ter*=`1`, *model*=`0`, *end*=`0`, *info*=`0`, *HSkip*=`0`, *allModels*=`0`, *verbose*=`0`)

PDB.out write a PDB

Parameters

`outName:`	the name of the file written, if none it sets the standard out
`chainId:`	
`altCare:`	does the file contains the alternate atoms (Yes:1 by default)
`altLbl:`	does the file contains the alternate label (Yes:1 by default)
`OXTCare:`	does the file contains the information of each line (Yes:1 by default)
`hetSkip:`	does the file skip the het (No:0 by default)
`fmode:`	the opening file method, ”w” writing by default
`header:`	does the file contains the header (Yes:1 by default)
`ter:`	does the file keep TER lines (Yes:1 by default)
`model:`	the number of the model that you want to write in the file
`end:`	does the file keep END line (No:0 by default)
`info:`	does the file contains the information of each line (Yes:1 by default)
`HSkip:`	does the file skip the hydrogen (No:0 by default)
`allModels:`	does the file contains all the models existing (No:0 by default)

Return Value

none, it outputs PDB content to file

Overrides: PyPDB10.atmList.out

xyzout(*self*, *outName*=`''`, *chainId*=`''`, *hetSkip*=`0`, *fmode*=`'w'`, *verbose*=`0`)

xyzout writes atoms coordinates in a file

Parameters

`outName`: the name of the file written, if none it sets the standard out

`fmode`: the opening file method, ”w” writing by default

Return Value

none, the file ”outName” is written

xyz(*self*, *outName*=`''`, *chainId*=`''`, *hetSkip*=`0`, *fmode*=`'w'`, *verbose*=`0`)

xyz

Parameters

`outName`: the name of the file written, if none it sets the standard out

`chainId`:

`hetSkip`:

`fmode`: the opening file method, ”w” writing by default

Return Value

the coordinates of all the atoms, they are concatenated

Overrides: PyPDB10.atmLine.xyz

loadXML(*self*, *fname*, *chainId*=`''`, *hetSkip*=`0`, *PDBDIR*=`'/data/pdb/data/structures/'`, *CATHDIR*=`'/data/banks/CATH/current/pdb'`, *SCOPDIR*=`'/data/banks/Astral/current/'`, *verbose*=`0`, *model*=`1`)

---

loadXML currently converts atoms informations in flat and parses them. You must add "isXML=True" to load a XML file

Parameters

`fname`:	the name of the file
`chainId`:	the Chain id
`hetSkip`:	does the file skip the hetero atoms (No:0 by default)
`PDBDIR`:	equal to GDFLTPDBDIR
`CATHDIR`:	equal to GDFLTCATHDIR
`SCOPDIR`:	equal to GDFLTSCOPDIR

Return Value

the atoms informations parsed

Note: it does not support/try cath, scop and astral

load(*self*, *fname*, *chainId*=`''`, *hetSkip*=`0`, *PDBDIR*=`'/data/pdb/data/structures/'`, *CATHDIR*=`'/data/banks/CATH/current/pdb'`, *SCOPDIR*=`'/data/banks/Astral/current/'`, *verbose*=`0`, *model*=`1`)

---

load parse a PDB file

Parameters

`fname`:	the name of the file
`chainId`:	the Chain id
`hetSkip`:	does the file skip the hetero atoms (No:0 by default)
`PDBDIR`:	equal to GDFLTPDBDIR
`CATHDIR`:	equal to GDFLTCATHDIR
`SCOPDIR`:	equal to GDFLTSCOPDIR

Return Value

the atoms informations parsed

parse(*PDB*, *self*, *allPDB*, *id*=`""`, *chainId*=`""`, *hetSkip*=`0`, *verbose*=`0`, *model*=`1`)

Parameters

`allPDB`: the flat lines to format

`id`: the id of PDB

`chainId`: the Chain id

`hetSkip`: does the file skip the hetero atoms (No:0 by default)

`model`: the model sets as working set (1 by default)

Return Value

a PDB format

resTab(*self*, *verbose*)

PDB.resTab reformats the data for an easier access

Return Value

none

Note: update PDB.rt atoms from PDB.atms atoms

atmTab(*self*)

PDB.atmTab reformats the data for an easier access

Return Value

none

Author: F.Briand

Note: update PDB.atms atoms from PDB.rt atoms

nModels(*PDB*)

Return Value

number of models of PDB instance (as defined by MODEL / ENDMDL lines)

setModel(*self*, *model*=`1`, *verbose*=`0`)

This will install the model of rank specified by ”model” as the current working set. PDB.setModel(model = 1,verbose = 0)

Parameters

`model`: the number of the model you want to set as working model

Return Value

none

chnList(*PDB*)

Return Value

a string, the concatenated chain identifiers present in the PDB (including blank).

nChn(*PDB*)

Return Value

a number, the number of chain identifiers present in the PDB (including blank).

hasChn(*self*, *chnId*)

PDB.hasChn check if chain of id chainId is present in the PDB instance.

Parameters

`chnId`: the id of a chain

Return Value

number of chnId in the list of chain

chn(*PDB*, *chnId*, *hetSkip*=`0`)

Parameters

`chainId`: the id of a chain

(type=might contain several chain Ids (e.g. AB))

`hetSkip`: does the file skip the hetero atoms (No:0 by default)

`inplace`: does the chn function will replace 'self' by the truncated PDB (No:0 by default)

Return Value

a PDB instance of chains of the PDB.

Author: F.Briand

Note: if chainId starts by "-" (minus) returns all but the chains specified.

chnRename(*self*, *pattern*=`''`, *verbose*=`0`)

PDB.chnRename Rename chains

Parameters

`pattern:` ”chain1chain2:newChain1newChain2”

Return Value

none

Author: F.Briand

Note: edit the PDB class ”in place”, ”:newChain” give a name to a ”one chain PDB”

renameHydrogens(*self*, *norm*, *verbose*=`0`)

PDB.renameHydrogens

Parameters

`norm:`	IUPAC to set Hydrogens to IUPAC norm, PDB to set Hydrogens to PDB norm
`verbose:`	set verbose mode

Return Value

none

Author: F.Briand

Note: It only works for standard amino-acids.

hNorm(*self*)

PDB.hNorm give the hydrogen naming convention of the PDB.

Author: F.Briand @return : the norm in {lIUPAC, lPDB, lpIUPAC, lpPDB, None} @note : ”l” means legacy, ”p” means ”partial”, None if norms are ”indifferentiable”

chnType(*self*, *chainId*=`''`, *verbose*=`0`)

PDB.chnType give the molecular type of chain(s)

Parameters

`chainId:` the chain id

Return Value

Heuristic detection if the chain type is one of:

- Protein
- RNA
- DNA
- SOLVENT
- HETERO

resTypes(*self*, *what*=`'all'`, *types*=`'all'`, *solvent*=`1`)

PDB.resTypes return a list of the residue names

Parameters

`what:` could be "all" or "aminoacid" or "nucleicacid"

`types:` could be "all" or "none" or "std" or "lstd" or "het" or "shet"

`solvent:` consider solvent (true by default)

Return Value

a list of residue names (3 characters), combining mask values of what and solvent

Note:

- "all" : (default) all residue types.
- "none": no residue types (only solvent mask is effective).
- "std" : all standard residue types (standard amino acids, standard nucleotides).
- "lstd": all amino acid types in addition to std.
- "het" : all non standard residues (includes non standard amino-acids)
- "shet": true heteros groups (does not include non standard amino-acids)

select(*self*, *rwhat*=`['']`, *awhat*=`['']`)

PDB.select return a selection of (sub) residues

Parameters

`rwhat:` none, or what residues

`awhat:` none, or what atoms

Return Value

a selection of (sub) residues

Overrides: PyPDB10.residue.select

mask(*self*, *ffrom*=`0`, *tto*=`-1`, *mask*=`''`)

PDB.mask return a selection of (sub) residues for a structure

Parameters

`ffrom:` the first (sub) residues of the selection

`tto:` the last (sub) residues of the selection

`mask:` (if specified) is a string of length to-from, positions corresponding to '-' will be discarded

Return Value

a selection of (sub) residues for a structure

header(*self*)

PDB.HEADER

Parameters

`hdr:` none, or a new header

Return Value

the title of the file

Overrides: PyPDB10.atmLine.header

compound(*self*)

PDB.compound

Return Value

the nature of the file (string)

source(*self*)

PDB.source

Return Value

where the molecule come from (string)

author(*self*)

PDB.author

Return Value

the author of the structure

keywords(*self*)

PDB.keywords

Return Value

a list of keywords

date(*self*)

PDB.date

Return Value

creation date of the file (a string)

revdate(*self*)

PDB.revdate (supposes last revision is first REVDAT)

Return Value

a string coding for the last revision date.

expmethod(*self*, *verbose*=`0`)

PDB.expmethod

Return Value

method by which crds were generated, it corresponds to the values of the EXPDTA field:'X-RAY DIFFRACTION', 'NMR', 'ELECTRON DIFFRACTION', etc.

resolution(*PDB*)

Return Value

the Resolution of the file, if specified somewhere (return -1 if not found). (data mining in the REMARK lines)

Note: This is only available for files determined using Xray.

rvalue(*PDB*)

A corresponding method is defined for the free R value:

Return Value

the RValue of the file (float), if specified somewhere (-1 if not). (data mining in the REMARK lines)

freervalue(*self*)

PDB.freervalue() look for the Rvalue

Return Value

the rvalue or NULL if not found

seqresaa3(*PDB*)

Parameters

`chIds:` the chains Ids

Return Value

the sequence of the PDB as specified in the SEQRES lines.

seqres(*PDB*)

Parameters

`chIds:` It is possible to specify chain Id

Return Value

a string of the sequence in the SEQRES lines. If several chains exist:
a list of all the sequences is returned.

Note: *example:* x.seqres("ABD") will return a list of the three sequences (if exist) corresponding to the chains A, B and D.

CAonly(*PDB*)

Return Value

Does the file contain only CAs ? (Yes/No)

SCatmMiss(*PDB*)

Return Value

- nSCMiss: the number of amino-acid residues having some side chain missing atom
- SCatmMiss: a string containing the information about all the residues with missing side chain atoms? (Yes/No)

Notes:

- For each residue, the string (SCatmMiss) consists of RName_ChLbl_RNum_RIcode, where RName is the name of the residue (3 letters), ChLbl is the chain label, RNum is the number of the residue (as in the PDB file) and RIcode the PDB insertion code of the residue.
- For residues having at least one atomic coordinate present

BBatmMiss(*PDB*)

Return Value

- nBBmiss: the number of amino-acid residues having some backbone missing atom (one of N, CA, C, O)
- BBatmMiss: nd a string concatening the information about all the residues with missing peptidic chain atoms

Notes:

- For each residue, the string (BBatmMiss) consists of RName_ChLbl_RNum_RIcode, where RName is the name of the residue (3 letters), ChLbl is the chain label, RNum is the number of the residue (as in the PDB file) and RIcode the PDB insertion code of the residue.
- For residues having at least one atomic coordinate present

hasAltAtms(*self*, *verbose*=`0`)

PDB.hasAltAtms

Return Value

Does the file has BBaltAtm or SCAltAtm? (Yes/No for each)

Overrides: PyPDB10.residue.hasAltAtms

altAtmsResList(*self*, *verbose*=`0`)

PDB.altAtmsResList

Return Value

nBBAltAtm, BBAltAtm, nSCAltAtm, SCAltAtm:

- nBBAltAtm: number of Back Bones in alternate atoms
- BBAltAtm: the Back Bones in alternate atoms
- nSCAltAtm: number of side chain in alternate atoms
- SCAltAtm: side chain in alternate atoms

geomCheck(*self*, *verbose*=`0`)

PDB.geomCheck() This will scan and check that the peptidic bonds geometry is rather correct. It is based on the value of the peptidic bond.

Return Value

Is the BB peptidic geometry (distance) correct? (OK/Poor/Bad)

Note: THIS WILL NOT DETECT FRAGMENTS. IF MANY, THE GAPS ARE IGNORED AND DO NOT RESULT IN ”Bad” RETURN.

This allows to scan that all the fragments are correct at once.

traceCheck(*self*, *hetSkip*=`0`, *maxCADist*=`4.2`, *verbose*=`0`)

PDB.traceCheck check if BB peptidic geometry is correct (distance)

Parameters

`maxCADist`: the maximum distance between 2 CA consecutive, 4.2 angstrom by default

Return Value

traceOK (OK/bad), tracePB (residues with bad geometry), nCISPRO (number of cis prolines), CISPRO (cis prolines), nCISPep (number of cis peptides), CISPep (cis peptides)

traceCheck2(*self*, *hetSkip*=`0`, *minCADist*=`3.7`, *maxCADist*=`3.9`, *verbose*=`0`)

PDB.traceCheck2 check if BB peptidic geometry is correct (distance)

Parameters

`minCADist`: the minimum distance between 2 CA consecutive, 3.7 angstrom by default

`maxCADist`: the maximum distance between 2 CA consecutive, 3.9 angstrom by default

Return Value

traceOK (OK/bad), tracePB (residues with bad geometry), nCISPRO (number of cis prolines), CISPRO (cis prolines), nCISPep (number of cis peptides), CISPep (cis peptides)

resLabel(*PDB*)

Parameters

`aRes`: a residue

Return Value

the label of the residue

traceCheck3(*self*, *hetSkip*=`0`, *minCADist*=`3.7`, *maxCADist*=`3.9`, *verbose*=`0`)

PDB.traceCheck3 check if BB peptidic geometry is correct (distance)

Parameters

`minCADist`: the minimum distance between 2 CA consecutive, 3.7 angstrom by default

`maxCADist`: the maximum distance between 2 CA consecutive, 3.9 angstrom by default

Return Value

traceOK (OK/bad), tracePB (residues with bad geometry), nCISPRO (number of cis prolines), nCISPep (number of cis peptides)

CISSeq(*self*, *hetSkip*=`0`, *minCADist*=`3.7`, *maxCADist*=`3.9`, *verbose*=`0`)

CISSeq

Parameters

`minCADist`: the minimum distance between 2 CA consecutive, 3.7 angstrom by default

`maxCADist`: the maximum distance between 2 CA consecutive, 3.9 angstrom by default

`hetSkip`: does the file skip the hetero atoms (No:0 by default)

Return Value

the prolines and other peptides found to be in the cis conformation.

chnCAFrgList(*self*, *chId*=`''`, *maxDist*=`4.1`)

PDB.chnCAFrgList determine fragments based on alpha carbon inter-atomic distance alone

Parameters

`chId`: a chain ID

`maxDist`: the maximal distance between two consecutive AC to be in the same fragment (default = 4.10)

Return Value

the chain with fragments separated and the number of fragments.

asOneChn(*self*, *chnId*=`' '`)

PDB.asOneChn

Parameters

`chnId`: a chain ID

Return Value

a PDB instance with residues renumbered as if there were only one chain.

resRenumber(*self*, *pattern*=`''`, *verbose*=`0`)

PDB.resRenumber Renumber residus

Parameters

`pattern`: "chnName1(string):ffrom1(integer):tto1(integer):index1(integer) chnName2(string):ffrom2(integer):tto2(integer):index2(integer) ..."

Return Value

none

Author: F.Briand

Note: edit the PDB class "in place"

atmsRenumber(*self*, *pattern*=`''`, *verbose*=`0`)

PDB.atmsRenumber Renumber atoms

Parameters

`pattern`: index(integer):ffrom(integer):tto(integer)

Return Value

True, False if interrupt (indexation problem)

Author: F.Briand

Note: edit the PDB class "in place"

resMeanBVal(*self*, *pattern*=`''`, *verbose*=`0`)

PDB.resMeanBVal Renumber the B Value with the mean on each residue

Parameters

`pattern`: chn1:ffrom1:tto1_chn2:ffrom2:tto2_...

Return Value

Nothing, edit "in place"

Author: F.Briand

atmsBValRenumber(*self*, *input*, *verbose*=`0`)

PDB.atmsBValRenumber renumber B Values according to the pattern in input

Parameters

`input`: ”newBValue” for all the atoms or pairs ”AtomNumber newBValue”. 1 BValue per line, separated by blank (tabs, spaces ...)

Return Value

Nothing, edit ”in place”

Author: F.Briand

chnFrgList(*self*, *chId*=`''`, *maxDist*=`1.7`)

PDB.chnFrgList determine fragments based on inter-atomic distance C’-N alone

Parameters

`chId`: a chain ID

`maxDist`: the maximal C’-N distance to be in the same fragment

Return Value

the chain with fragments separated and the number of fragments.

Note: 1.70 is default threshold

frgList(*self*, *maxNCDist*=`1.7`, *maxCADist*=`4.1`, *verbose*=`0`)

PDB.frgList will return the number of fragments and their boundaries

Parameters

`maxNCDist`: the maximum distance between N and C to be in the same fragment

`maxCADist`: the maximum distance between 2 consecutive AC to be in the same fragment

Return Value

a list of the fragments of the PDB if some geometric inconsistencies are detected and the numbers of fragments

nFrg(*self*, *maxNCDist*=`1.7`, *maxCADist*=`4.1`, *verbose*=`0`)

nFrg

Parameters

`maxNCDist`: the maximum distance between N and C to be in the same fragment

`maxCADist`: the maximum distance between 2 consecutive AC to be in the same fragment

Return Value

the numbers of fragments detected based on NC and CA distances

aaseq_ori(*PDB*)

Return Value

the sequence of residues present in the PDB file, having coordinates.

Note: Converts non standard amino-acids to equivalent standard amino-acid.

aaseq(*PDB*)

Parameters

`matchAtms`: a list of atoms searched

Return Value

a list of atoms matching

frgseq(*self*, *maxNCDist*=`1.7`, *maxCADist*=`4.1`, *verbose*=`0`)

PDB.frgseq fragments the sequence according to maxNCDist and maxCADist

Parameters

`maxNCDist`: the maximum distance between N and C to be in the same fragment

`maxCADist`: the maximum distance between 2 consecutive AC to be in the same fragment

Return Value

the fragments of the PDB if some geometric inconsistencies are detected and the numbers of fragments

SGList(*self*)

PDB.SGList

Return Value

a list of all the coordinates of the gamma sulfur

nSSIntra()

Return Value

nSSbond the number of SSbond in a protein

SSIntra()

Return Value

the position of the cysteins involed in a SSbond

isHalfCys(*self*, *aRes*)

isHalfCys(aRes) checks if the distance between the sulfur of two cysteins allows a disulfide bond

Parameters

`aRes:` the number of the residue, must be a "CYS"

Return Value

the position of the second cys and the distance separating them

findRes(*self*, *chId*, *rName*, *rNum*, *icode*, *what*=`None`, *verbose*=`0`)

PDB.findRes To identify a residue given its chain Id, name, PDB number, insertion code

Return Value

- either the residue if what == None
- or the residue rank (from 0) if what != None

findAtm(*self*, *chId*, *rName*, *rNum*, *icode*, *atmName*=`'CA'`, *verbose*=`0`)

PDB.findAtm To identify an atom given residue chain Id, name, PDB number, insertion code and atom Name

Parameters

`chId:`	the chain Id
`rName:`	the residu name
`rNum:`	the residu number
`icode:`	the line code number
`atmName:`	the atom name
`verbose:`	if verbose=1 it will print the aAtm loop or None if not found

Return Value

either the atom instance

Overrides: PyPDB10.residue.findAtm

clean_ori(*whatRes*=`"5HP"`, *"PCA"*=`['PCA', '5HP', 'FGL', 'MSE', 'CSE', 'CEA', 'CGU', 'HTR', ...`, *"MSE"*=`0`, *"CSE"*, *"CEA"*, *"CGU"*, *"HTR"*, *"TPQ"*)

---

cleanup PDB files by converting some non standard residue into standard ones.

Parameters

`whatRes`: it is the list of residues that may be affected.

(type=the default is all. whatRes could be only part of the default list.)

Note: + transformation des residus PCA en GLU + transformation des residus MHO en MET + transformation des residus IAS en ASP + transformation des residus HYP en PRO + transformation des residus TPQ en TYR + transformation des residus TRO en TRP + transformation des residus TYS en TYR + transformation des residus MSE en MET

- transformation des atomes SE en S

+ transformation des residus CSE en CYS

- transformation des atomes SE en S

+ transformation des residus CEA en CYS

- suppression des atomes O1 et HO1

+ transformation des residus CGU en GLU

- suppression des atomes CD2, OE3, OE4, HE4

+ transformation des residus HTR en TRP

- suppression des atomes O et OH

+ transformation des residus TPQ en PHE

- suppression des atomes O2, O4, O5 et HO4

+ transformation des residus FGL en SER

- suppression des atomes OG1, renomme OG1 en OG

+ transformation des residus AYA (acetyl ALA) en ALA

- suppression des atomes CT, OT, CM

+ transformation des residus FME (formyl MET) en MET

- suppression des atomes OF, CF, (also CN, O1, HCN)

+ transformation des residus CXM (carboxy MET) en MET

- suppression des atomes CN, O1, O2, HO1, HO2

+ transformation des residus SAC (acetyl SER) en SER

- suppression des atomes C1A, C2A, OAC, 1H2A, 2H2A, 3H2A

+ transformation des residus CSO (s-hydroxycysteine) en CYS

- suppression des atomes OD, HD

+ transformation des residus BET (3methyl GLY) en GLY (NOT BY DEFAULT)

- suppression des atomes C1, C2, C3, 1H1, 1H2, 1H3, 2H1, 2H2, 2H3, 3H1, 3H2, 3H3

clean(*whatRes*=`"5HP"`, *"PCA"*=`['PAQ', 'AGM', 'PR3', 'DOH', 'CCS', 'GSC', 'GHG', 'OAS', ...`, *"MSE"*=`0`, *"CSE"*, *"CEA"*, *"CGU"*, *"HTR"*, *"TPQ"*)

Parameters

`whatRes`: it is the list of residues that may be affected.

(type=the default is all. whatRes could be only part of the default list.)

Note: cleanup PDB files by converting some non standard residue into standard ones. + transformation des residus PCA en GLU + transformation des residus MHO en MET + transformation des residus IAS en ASP + transformation des residus HYP en PRO + transformation des residus TPQ en TYR + transformation des residus TRO en TRP + transformation des residus TYS en TYR + transformation des residus MSE en MET

- transformation des atomes SE en S

+ transformation des residus CSE en CYS

- transformation des atomes SE en S

+ transformation des residus CEA en CYS

- suppression des atomes O1 et HO1

+ transformation des residus CGU en GLU

- suppression des atomes CD2, OE3, OE4, HE4

+ transformation des residus HTR en TRP

- suppression des atomes O et OH

+ transformation des residus TPQ en PHE

- suppression des atomes O2, O4, O5 et HO4

+ transformation des residus FGL en SER

- suppression des atomes OG1, renomme OG1 en OG

+ transformation des residus AYA (acetyl ALA) en ALA

- suppression des atomes CT, OT, CM

+ transformation des residus FME (formyl MET) en MET

- suppression des atomes OF, CF, (also CN, O1, HCN)

+ transformation des residus CXM (carboxy MET) en MET

- suppression des atomes CN, O1, O2, HO1, HO2

+ transformation des residus SAC (acetyl SER) en SER

- suppression des atomes C1A, C2A, OAC, 1H2A, 2H2A, 3H2A

+ transformation des residus CSO (s-hydroxycysteine) en CYS

- suppression des atomes OD, HD

+ transformation des residus BET (3methyl GLY) en GLY (NOT BY DEFAULT)

- suppression des atomes C1, C2, C3, 1H1, 1H2, 1H3, 2H1, 2H2, 2H3, 3H1, 3H2, 3H3

site(*self*, *verbose*=`0`)

Parse the info lines and check for a site description according to the PDB format

Return Value

a dictionnary or None

CSAsite(*self*, *id*=`None`, *verbose*=`0`)

PDB.CSAsite

Parameters

`id:` a PDB id

Return Value

a list of dictionnaries

Note:

- Attempt to retrieve site from Catalytic Site Atlas:

at http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/CSA/CSA_Site_Wrapper.pl?pdb=2lzm returns either None or a dictionnary of the sites

- CSA does not consider chains. Hence, one must parse it later.
- Status: ”Literature” or ”PsiBLAST”
- Referer: ”PsiBLAST” match
- Comment: Comment on psiBlast and EC.
- Site: Atoms involved

exposedAminoAcids(*self*, *rH2O*=`'1.4'`, *ASALimit*=`'0.25'`, *what*=`'E'`, *verbose*=`0`)

PDB.exposedAminoAcids: determine if an amino acid is exposed to the solvant

Parameters

`what:`	
`rH2O:`	the radius of the H2O molecule
`ASALimit:`	exposure threshold beyond which the residue is considered as exposed

Return Value

a PDB instance with all the residues exposed to the solvant

BB(*PDB*)

Return Value

a PDB of the backbone atoms only

SC(*PDB*)

Return Value

a PDB of the side-chain atoms only

around(*self*, *elt*, *aPos*, *dist*=`3.0`, *verbose*=`0`)

PDB.around search all the atoms or residus around aPos to a distance dist

Parameters

`elt:` ”a” or ”r”, respectively for atoms or residus

`dist:` distance in angstrom

`aPos:` atom position

Return Value

none, print the name and number of atoms around the atom position

addHydrogens(*self*, *algorithm*=`'HAAD'`, *HSkip*=`1`, *norm*=`None`, *verbose*=`0`)

```
PDB.addHydrogens add Hydrogens to the current structure using HAAD (Li,
    et al.(2009) "HAAD: A Quick Algorithm for Accurate Prediction of
    Hydrogen Atoms in Protein Structures" PLoS One, 4: e6701.) or Reduce
    (Word, et al.(1999) "Asparagine and glutamine: using hydrogen atom
    contacts in the choice of sidechain amide orientation" J. Mol. Biol.
    285, 1735-1747) methods
@author: F.Briand
@param method: Method used (HAAD, Reduce)
@param HSkip: Does the hydrogens already in place are skipped or not ? (True/False)
@param norm: Which norm have to be applied ? None (keep norm of the algorithm), lIUP
@param verbose: if > 0, step avancement is printed; if > 1, CONECT fields changes a
@return: None
@note: Hydrogens are added "in place"
```

atmsForceRenumber(*self*, *verbose*=`0`)

PDB.atmsForceRenumber renumber all the atoms of a PDB structure, starting to the first atom with index equal to its atom number. Also increase the index by 1 on TER lines.

Parameters

`verbose`: if > 0, step avancement is printed; if > 1, CONECT fields changes are printed

Return Value

None

Author: F.Briand

Note: renumbering ”in place”

delHydrogens(*self*, *verbose*=`0`)

PDB.delHydrogens delete Hydrogens of a PDB structure then renumber atoms

Parameters

`verbose`: if > 0, step avancement is printed; if > 1, CONECT fields changes are printed

Return Value

None

Author: F.Briand

Note: Hydrogens are deleted ”in place”

trace(*self*, *hetSkip*=`0`, *verbose*=`0`)

PDB.trace:

Parameters

`hetSkip`: does hetero atom are skipped.0 skip nothing, 1 skip no-amino acids atoms, 2 skip no-amino acids & ”special” amino acids.

Return Value

a PDB of the CA

getAtoms(*self*, *pattern*=`''`, *byID*=`0`, *verbose*=`0`)

PDB.getAtoms extract a list of atoms from a PDB structure

Parameters

`pattern`: "ffrom:tto", tto not included

`byID`: does the function select atoms by python list index (False), or by atom number (True).

Return Value

a PDB structure

Author: F.Briand

getMultiAtoms(*self*, *pattern*=`''`, *verbose*=`0`)

PDB.getMultiAtoms extract lists of atoms from a PDB structure

Parameters

`pattern`: "from1:to1 from2:to2 ...", to not included @return a PDB structure

Author: F.Briand

addOXT(*self*, *chainId*=`''`, *verbose*=`0`)

PDB.addOXT add terminal oxygens

Parameters

`chainId`: set chains which have their OXT to be added. "" = all the chains

Author: F.Briand

rmk350apply(*self*, *biomolecule*=`None`, *verbose*=`0`)

PDB.rmk350apply apply the translation/rotation matrix which is in REMARK 350 fields

Return Value

a PDB with the matrix applied

Author: F.Briand

checkBBOrder(*self*, *verbose*=`0`)

PDB.checkBBOrder check that backbone atoms order is : N, CA, C, O, (OXT) and restore correct order if it wasn't.

Author: F.Briand

Inherited from PyPDB10.residue(Section 1.6)

BBAtmMiss(), altLbls(), atmPos(), chnLbl(), delete(), rName(), rNum(), rType(), riCode(), setBBOrder()

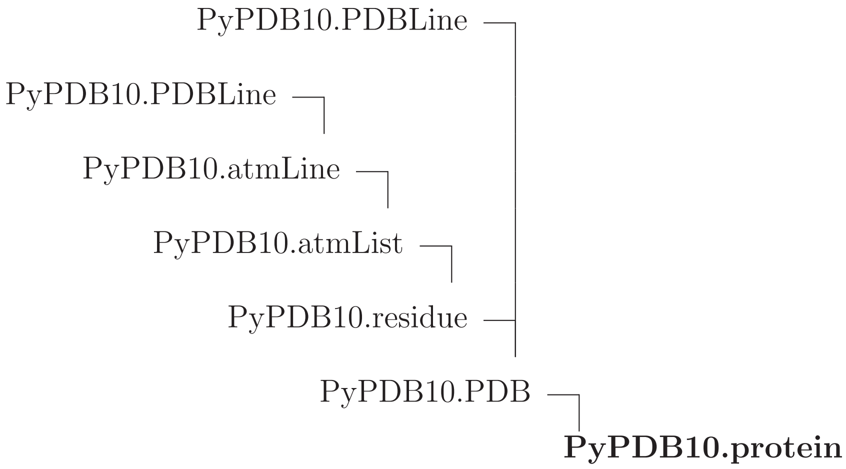
Inherited from PyPDB10.atmList(Section 1.5)

BC(), CApos(), Cpos(), Npos(), Opos(), __setitem__(), chis(), crds(), insert(), isPDB(), oneChis(), oneHMMGeo(), outChis(), radius(), resName(), theAtm(), write()

Inherited from PyPDB10.atmLine(Section 1.4)

alt(), atmBVal(), atmName(), atmNum(), atmType(), chrg(), ele(), fpt(), icode(), occ(), q(), r(), resNum(), resType(), segId(), setcrds(), tfac()

1.8 Class protein



class protein This models protein

1.8.1 Methods

__init__(*self*, *data*, *chId*=`''`, *model*=`1`, *hetSkip*=`0`, *verbose*=`0`)

PDB.__init__ determine the type of data to initialize

Parameters

`data:`	an instance
`chId:`	a chain Id
`model:`	the number of the model you want to set as working model (number 1 by default)
`hetSkip:`	does the file skip the hetero atoms (No:0 by default)

Return Value

none

Overrides: PyPDB10.atmLine.__init__

resTypes(*self*, *verbose*=`0`)

PDB.resTypes

Parameters

`what:`	could be ”all” or ”aminoacid” or ”nucleicacid”
`types:`	could be ”all” or ”none” or ”std” or ”lstd” or ”het” or ”shet”
`solvent:`	consider solvent (true by default)

Return Value

a list of restypes of the protein

Overrides: PyPDB10.PDB.resTypes

frgList(*PDB*)

PDB.frgList will return the number of fragments and their boundaries

Parameters

`maxNCDist`: the maximum distance between N and C to be in the same fragment

`maxCADist`: the maximum distance between 2 consecutive AC to be in the same fragment

Return Value

a list of the fragments of the PDB if some geometric inconsistencies are detected and the numbers of fragments

Overrides: PyPDB10.PDB.frgList

Note: the detection of fragments is based on the NC distance

nFrgs(*self*)

nFrgs

Return Value

the number of fragments

Note: the detection of fragments is based on the NC distance

trace(*self*, *fname*=`''`, *chId*=`''`, *hetSkip*=`0`, *altSel*=`' '`)

PDB.Trace:

Parameters

`fname`: the file name.

`hetSkip`: does the file skip the hetero atoms (No:0 by default)

Return Value

an atom list of the CA

Overrides: PyPDB10.PDB.trace

outSeq(*protein*)

Parameters

`hetSkip`: does the file skip the het (No:0 by default)

Return Value

none, it prints the trace of the protein sequence

outRawSeq(*protein*)

Parameters

`hetSkip`: does the file skip the het (No:0 by default)

Return Value

none, it prints the trace of the protein sequence

aaseq(*PDB*)

Parameters

`matchAtms`: a list of atoms searched

Return Value

the sequence of residues present in the PDB file, having coordinates.

Overrides: PyPDB10.PDB.aaseq

Note: Converts non standard amino-acids to equivalent standard amino-acid.

frg(*self*, *whatFrg*, *frgs*=`[]`)

PDB.frg

Parameters

`whatFrg`: the fragment of the protein we need

`frgs`: a list of fragments

Return Value

the atoms of the fragment ”whatFrg”

hasAltAtms(*self*, *verbose*)

PDB.hasAltAtms This will return 2 values consisting of “Yes” or “No”. The first answers the question: does some x amino-acid backbone atoms have alternate coordinates (as specified in PDB files). The second anwsers the corresponding question for side chains.

Return Value

Does the file has BBaltAtm or SCAltAtm? (Yes/No for each)

Overrides: PyPDB10.residue.hasAltAtms

altAtmsResList(*self*, *verbose*)

PDB.altAtmsResList This function is related to "hasAltAtms". It will, for backbone and side chains return the number of residues having backbone alt coordinates followed by a string containing the information about these residues (in a format similar to that described for SCatmMiss. The two first values concern backbone, the two next side chains.

Return Value

nBBAltAtm, BBAltAtm, nSCAltAtm, SCAltAtm:

- nBBAltAtm: number of Back Bones in alternate atoms
- BBAltAtm: the Back Bones in alternate atoms
- nSCAltAtm: number of side chain in alternate atoms
- SCAltAtm: side chain in alternate atoms

Overrides: PyPDB10.PDB.altAtmsResList

hasAllBBAtms(*self*, *verbose*)

hasAllBBatms checks if all BB atoms are present

Return Value

the position of the BB atoms missing

geomCheck(*PDB*)

PDB.geomCheck() This will scan and check that the peptidic bonds geometry is rather correct. It is based on the value of the peptidic bond.

Return Value

Is the BB peptidic geometry (distance) correct? (OK/Poor/Bad)

Overrides: PyPDB10.PDB.geomCheck

Note: THIS WILL NOT DETECT FRAGMENTS. IF MANY, THE GAPS ARE IGNORED AND DO NOT RESULT IN "Bad" RETURN.

This allows to scan that all the fragments are correct at once.

traceCheck(*self*, *hetSkip*=`0`, *verbose*=`0`)

PDB.traceCheck check if BB peptidic geometry is correct (distance)

Parameters

`hetSkip`: does the file skip the het (No:0 by default)

Return Value

traceOK (OK/bad), tracePB (residues with bad geometry), nCISPRO (number of cis prolines), CISPRO (cis prolines), nCISPep (number of cis peptides), CISPep (cis peptides),CisWarning (CisPRO/CisPEP), hasCisPRO(Yes/No), hasCisPEP(Yes,No)

Overrides: PyPDB10.PDB.traceCheck

BBAngles(*self*, *aRes*=`-1000`)

PDB BBangles calculate phi psi ome of 2 consecutives residues

Parameters

`aRes`: a residue number, -1000 by default in this case it will calculate all phi psi ome of BB angles.

Return Value

angles phi psi ome, or a list of them if aRes is not set by user.

SGList(*self*)

PDB.SGList

Return Value

a list of all the coordinates of the gamma sulfur

Overrides: PyPDB10.PDB.SGList

nSSIntra()

Return Value

the number of SSbonds in the PDB instance

Overrides: PyPDB10.PDB.nSSIntra

BB(*protein*)

Return Value

a protein with the backbone atoms only

Overrides: PyPDB10.PDB.BB

SC(*protein*)

Return Value

a protein with the side-chains atoms only

Overrides: PyPDB10.PDB.SC

Inherited from PyPDB10.PDB(Section 1.7)

BBatmMiss(), CAonly(), CISSeq(), CSAsite(), SCatmMiss(), SSIntra(), __add__(), __delitem__(), __getitem__(), __getslice__(), __len__(), __repr__(), aaseq_ori(), addHydrogens(), addOXT(), around(), asOneChn(), atmTab(), atmsBValRenumber(), atmsForceRenumber(), atmsRenumber(), author(), checkBBOrder(), chn(), chnCAFrgList(), chnFrgList(), chnList(), chnRename(), chnType(), clean(), clean_ori(), compound(), date(), delHydrogens(), expmethod(), exposedAminoAcids(), findAtm(), findRes(), flat(), freervalue(), frgseq(), getAtoms(), getMultiAtoms(), hNorm(), hasChn(), header(), isHalfCys(), keywords(), load(), loadXML(), mask(), nChn(), nFrg(), nModels(), out(), parse(), renameHydrogens(), resLabel(), resMeanBVal(), resRenumber(), resTab(), resolution(), revdate(), rmk350apply(), rvalue(), select(), seqres(), seqresaa3(), setModel(), site(), source(), traceCheck2(), traceCheck3(), xyz(), xyzout()

Inherited from PyPDB10.residue(Section 1.6)

BBAtmMiss(), altLbls(), atmPos(), chnLbl(), delete(), rName(), rNum(), rType(), riCode(), setBBOrder()

Inherited from PyPDB10.atmList(Section 1.5)

BC(), CApos(), Cpos(), Npos(), Opos(), __setitem__(), chis(), crds(), insert(), isPDB(), oneChis(), oneHMMGeo(), outChis(), radius(), resName(), theAtm(), write()

Inherited from PyPDB10.atmLine(Section 1.4)

alt(), atmBVal(), atmName(), atmNum(), atmType(), chrg(), ele(), fpt(), icode(), occ(), q(), r(), resNum(), resType(), segId(), setcrds(), tfac()

1.9 Class remark350

class remark350 this models the content of REMARK 350 fields

1.9.1 Methods

__init__(*self*, *input*=`''`, *verbose*=`0`)

remark350.__init__ initialize datas

Parameters

`input`: list of REMARK 350 lines

Author: F.Briand

__repr__(*self*)

remark350.__repr__

Return Value

return the content of instance (list of lines REMARK 350)

Author: F.Briand

__len__(*self*)

remark350.__len__

Return Value

number of REMARK 350 lines

Author: F.Briand

__getitem__(*self*, *rmkPos*)

remark350.__getitem__ return information calling self[rmkPos]

Parameters

`rmkPos`:
- "biomol" : list of biomolecules
- "biomt" : translation/rotation matrix
- "chains" : list of chains
- n (int) : n'ieme line of REMARK 350 fields

Return Value

Author: F.Briand

__getslice__(*self*, *ffrom*=`0`, *tto*=`None`)

remark350.__getslice__

Return Value

a slice of REMARK 350 lines : self[ffrom:tto]

Author: F.Briand

Index