

# Biomedical Informatics: Lecture 5

M. Ceccanti and A.Paoluzzi

Wed, Mar 26, 2014



# Worldwide Protein Data Bank (wwPDB)

The web site

consists of organizations that act as deposition, data processing and distribution centers for PDB data.

The founding members are

- **RCSB PDB (USA)**
- PDBe (Europe)
- and PDBj (Japan).

The **BMRB (USA)** group joined the wwPDB in 2006.

# Worldwide Protein Data Bank (wwPDB)

The web site

consists of organizations that act as deposition, data processing and distribution centers for PDB data.

The founding members are

- RCSB PDB (USA)
- PDBe (Europe)
- and PDBj (Japan).

The BMRB (USA) group joined the wwPDB in 2006.

# Worldwide Protein Data Bank (wwPDB)

The web site

consists of organizations that act as deposition, data processing and distribution centers for PDB data.

The founding members are

- RCSB PDB (USA)
- PDBe (Europe)
- and PDBj (Japan).

The BMRB (USA) group joined the wwPDB in 2006.

# Chemical Component Dictionary (CCD)

The CCD is as an external reference file describing all residue and small molecule components found in PDB entries

Search and browse the CCD using resources such as [PDBeChem](#) and [Ligand Expo](#).

- contains detailed chemical descriptions for standard and modified amino acids / nucleotides, small molecule ligands, and solvent molecules
- includes stereochemical assignments, aromatic bond assignments, idealized coordinates, chemical descriptors (SMILES & InChI), and systematic chemical names.
- is organized by the 3-character alphanumeric code that PDB assigns to each chemical component
- is updated with each weekly PDB release.

# Chemical Component Dictionary (CCD)

The CCD is as an external reference file describing all residue and small molecule components found in PDB entries

Search and browse the CCD using resources such as [PDBeChem](#) and [Ligand Expo](#).

- contains detailed chemical descriptions for standard and modified amino acids / nucleotides, small molecule ligands, and solvent molecules
- includes stereochemical assignments, aromatic bond assignments, idealized coordinates, chemical descriptors (SMILES & InChI), and systematic chemical names.
- is organized by the 3-character alphanumeric code that PDB assigns to each chemical component
- is updated with each weekly PDB release.

# Chemical Component Dictionary (CCD)

The CCD is as an external reference file describing all residue and small molecule components found in PDB entries

Search and browse the CCD using resources such as [PDBeChem](#) and [Ligand Expo](#).

- contains detailed chemical descriptions for standard and modified amino acids / nucleotides, small molecule ligands, and solvent molecules
- includes stereochemical assignments, aromatic bond assignments, idealized coordinates, chemical descriptors (SMILES & InChI), and systematic chemical names.
- is organized by the 3-character alphanumeric code that PDB assigns to each chemical component
- is updated with each weekly PDB release.



# Chemical Component Dictionary (CCD)

The CCD is as an external reference file describing all residue and small molecule components found in PDB entries

Search and browse the CCD using resources such as [PDBeChem](#) and [Ligand Expo](#).

- contains detailed chemical descriptions for standard and modified amino acids / nucleotides, small molecule ligands, and solvent molecules
- includes stereochemical assignments, aromatic bond assignments, idealized coordinates, chemical descriptors (SMILES & InChI), and systematic chemical names.
- is organized by the 3-character alphanumeric code that PDB assigns to each chemical component
- is updated with each weekly PDB release.

# The Protein Data Bank in Europe (PDBe)

## Bringing Structure to Biology

PDBe is the European resource for the collection, organisation and dissemination of data on biological macromolecular structures.

The main objectives of the work at PDBe are:

- to provide an integrated resource of high-quality macromolecular structures and related data and make it available to the biomedical community via intuitive user interfaces.
- to maintain in-house expertise in all the major structure-determination techniques (X-ray, NMR and EM) and on issues of mutual interest (such as data representation, formats and standards, or validation of structural data).
- to provide high-quality deposition and annotation facilities for structural data.

# The Protein Data Bank in Europe (PDBe)

## Bringing Structure to Biology

PDBe is the European resource for the collection, organisation and dissemination of data on biological macromolecular structures.

The main objectives of the work at PDBe are:

- to provide an integrated resource of high-quality macromolecular structures and related data and make it available to the biomedical community via intuitive user interfaces.
- to maintain in-house expertise in all the major structure-determination techniques (X-ray, NMR and EM) and on issues of mutual interest (such as data representation, formats and standards, or validation of structural data).
- to provide high-quality deposition and annotation facilities for structural data.

# The Protein Data Bank in Europe (PDBe)

## Bringing Structure to Biology

PDBe is the European resource for the collection, organisation and dissemination of data on biological macromolecular structures.

The main objectives of the work at PDBe are:

- to provide an integrated resource of high-quality macromolecular structures and related data and make it available to the biomedical community via intuitive user interfaces.
- to maintain in-house expertise in all the major structure-determination techniques (X-ray, NMR and EM) and on issues of mutual interest (such as data representation, formats and standards, or validation of structural data).
- to provide high-quality deposition and annotation facilities for structural data.

# European Bioinformatics Institute (EBI)

The European Bioinformatics Institute (EBI) is a non-profit academic organization that forms part of the European Molecular Biology Laboratory (EMBL)

The EBI is a centre for research and services in bioinformatics.

- The Institute manages databases of biological data including nucleic acid, protein sequences and macromolecular structures.
- For example: [Ligands in PDB](#)

# European Bioinformatics Institute (EBI)

The European Bioinformatics Institute (EBI) is a non-profit academic organization that forms part of the European Molecular Biology Laboratory (EMBL)

The EBI is a centre for research and services in bioinformatics.

- The Institute manages databases of biological data including nucleic acid, protein sequences and macromolecular structures.
- For example: [Ligands in PDB](#)

# European Bioinformatics Institute (EBI)

## Macromolecular Structure Database

- The MSD (Macromolecular Structure Database) group has moved to <http://www.ebi.ac.uk/pdbe/> and changed its name to the Protein Databank in Europe (PDBe).
- see [Macromolecular Structure Database Group Overview](#)
- A project is being undertaken to develop an Application Programming Interface (API) to the EBI-MSD database. This consists of a series of functions that will allow external 3rd party software to access the EBI-MSD database independently. This is based around a SOAP-XML based messaging system

# European Bioinformatics Institute (EBI)

## Macromolecular Structure Database

- The MSD (Macromolecular Structure Database) group has moved to <http://www.ebi.ac.uk/pdbe/> and changed its name to the Protein Databank in Europe (PDBe).
- see [Macromolecular Structure Database Group Overview](#)
- A project is being undertaken to develop an Application Programming Interface (API) to the EBI-MSD database. This consists of a series of functions that will allow external 3rd party software to access the EBI-MSD database independently. This is based around a SOAP-XML based messaging system



# European Bioinformatics Institute (EBI)

## Macromolecular Structure Database

- The MSD (Macromolecular Structure Database) group has moved to <http://www.ebi.ac.uk/pdbe/> and changed its name to the Protein Databank in Europe (PDBe).
- see [Macromolecular Structure Database Group Overview](#)
- A project is being undertaken to develop an Application Programming Interface (API) to the EBI-MSD database. This consists of a series of functions that will allow external 3rd party software to access the EBI-MSD database independently. This is based around a SOAP-XML based messaging system

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of **molecular databases**

- Services include:
  - **ENA** (DNA and RNA sequences)
  - **Ensembl** (genomes)
  - **ArrayExpress** (microarray data)
  - **UniProt** (protein sequences)
  - **PDB** (macromolecular structures)
  - **IntAct** (proteinprotein interactions)
  - **Reactome** (pathways)
  - **CiteXplore** (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same **principles of web service** provision

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of **molecular databases**

- Services include:
  - **ENA** (DNA and RNA sequences)
  - **Ensembl** (genomes)
  - **ArrayExpress** (microarray data)
  - **UniProt** (protein sequences)
  - **PDB** (macromolecular structures)
  - **IntAct** (proteinprotein interactions)
  - **Reactome** (pathways)
  - **CiteXplore** (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same **principles of web service** provision

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of **molecular databases**

- Services include:
  - **ENA** (DNA and RNA sequences)
  - **Ensembl** (genomes)
  - **ArrayExpress** (microarray data)
  - **UniProt** (protein sequences)
  - **PDB** (macromolecular structures)
  - **IntAct** (proteinprotein interactions)
  - **Reactome** (pathways)
  - **CiteXplore** (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same **principles of web service** provision

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of **molecular databases**

- Services include:
  - **ENA** (DNA and RNA sequences)
  - **Ensembl** (genomes)
  - **ArrayExpress** (**microarray** data)
  - **UniProt** (protein sequences)
  - **PDB** (macromolecular structures)
  - **IntAct** (proteinprotein interactions)
  - **Reactome** (pathways)
  - **CiteXplore** (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same **principles of web service** provision

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of **molecular databases**

- Services include:
  - **ENA** (DNA and RNA sequences)
  - **Ensembl** (genomes)
  - **ArrayExpress** (**microarray** data)
  - **UniProt** (protein sequences)
  - **PDB** (macromolecular structures)
  - **IntAct** (proteinprotein interactions)
  - **Reactome** (pathways)
  - **CiteXplore** (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same **principles of web service** provision

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of **molecular databases**

- Services include:
  - **ENA** (DNA and RNA sequences)
  - **Ensembl** (genomes)
  - **ArrayExpress** (**microarray** data)
  - **UniProt** (protein sequences)
  - **PDB** (macromolecular structures)
  - **IntAct** (proteinprotein interactions)
  - **Reactome** (pathways)
  - **CiteXplore** (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same **principles of web service** provision

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of [molecular databases](#)

- Services include:
  - [ENA](#) (DNA and RNA sequences)
  - [Ensembl](#) (genomes)
  - [ArrayExpress](#) ([microarray](#) data)
  - [UniProt](#) (protein sequences)
  - [PDB](#) (macromolecular structures)
  - [IntAct](#) (proteinprotein interactions)
  - [Reactome](#) (pathways)
  - [CiteXplore](#) (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same [principles of web service](#) provision



# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of [molecular databases](#)

- Services include:
  - [ENA](#) (DNA and RNA sequences)
  - [Ensembl](#) (genomes)
  - [ArrayExpress](#) ([microarray](#) data)
  - [UniProt](#) (protein sequences)
  - [PDB](#) (macromolecular structures)
  - [IntAct](#) (proteinprotein interactions)
  - [Reactome](#) (pathways)
  - [CiteXplore](#) (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same [principles of web service](#) provision

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of **molecular databases**

- Services include:
  - **ENA** (DNA and RNA sequences)
  - **Ensembl** (genomes)
  - **ArrayExpress** (**microarray** data)
  - **UniProt** (protein sequences)
  - **PDB** (macromolecular structures)
  - **IntAct** (proteinprotein interactions)
  - **Reactome** (pathways)
  - **CiteXplore** (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same **principles of web service** provision

# European Bioinformatics Institute (EBI)

EMBL-EBI maintains the worlds most comprehensive range of **molecular databases**

- Services include:
  - **ENA** (DNA and RNA sequences)
  - **Ensembl** (genomes)
  - **ArrayExpress** (**microarray** data)
  - **UniProt** (protein sequences)
  - **PDB** (macromolecular structures)
  - **IntAct** (proteinprotein interactions)
  - **Reactome** (pathways)
  - **CiteXplore** (EMBL portal to the scientific literature)
- The details of each database vary, but they all uphold the same **principles of web service** provision

# Drawing small molecules from PDB files

Include the Bio.PDB module from [Biopython](#) (and read [this tutorial](#) :o)

```
from Bio.PDB import *  
  
def myprint (string):  
    print "\n" + string + " ->", eval(string)
```

The `myprint()` function is used to show both an [expression](#) and the [value](#) produced by its [evaluation](#)

## Source program

The source of the following programming examples can be found in [pdb-ccd-example](#)

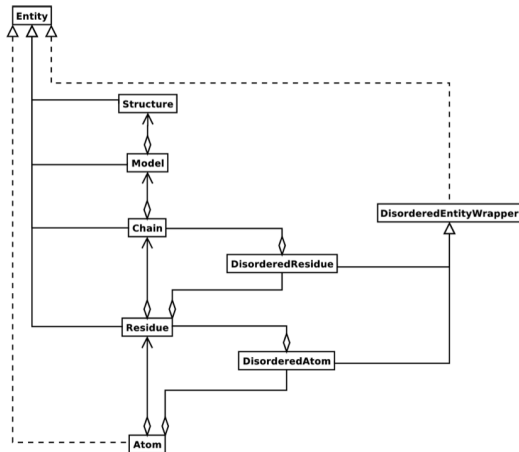
# Drawing small molecules from PDB files

The overall layout of a **Structure** object

The **Bio.PDB** package follows the **Structure/Model/Chain/Residue/Atom** architecture

- A structure consists of models
- A model consists of chains
- A chain consists of residues
- A residue consists of atoms

UML diagram of SMCRA architecture of the **Structure** object. Full lines with diamonds denote **aggregation**, full lines with arrows denote **composition**, full lines with triangles denote **inheritance** and dashed lines with triangles denote interface **realization**



# Get the input PDB file

One (of many) possible path ...

Search and browse the **Chemical Component Dictionary (CCD)** using resources such as PDBChem

- **Ligand Dictionary**
- **Ligand Expo**
  - 1 look at the short **Tutorial**
  - 2 and
    - 1 search
    - 2 browse
    - 3 download

# The input file BTC.pdb

```

HEADER      NONAME 22-Aug-09
TITLE       Produced by PDBeChem
COMPND      BTC
AUTHOR      EBI-PDBe Generated
REVDAT      1 22-Aug-09      0
ATOM        1  N   BTC      0      1.585   0.483  -0.081   1.00 20.00      N+0
ATOM        2  CA  BTC      0      0.141   0.450   0.186   1.00 20.00      C+0
ATOM        3  CB  BTC      0     -0.533  -0.530  -0.774   1.00 20.00      C+0
ATOM        4  SG  BTC      0     -0.247   0.004  -2.484   1.00 20.00      S+0
ATOM        5  C   BTC      0     -0.095   0.006   1.606   1.00 20.00      C+0
ATOM        6  O   BTC      0      0.685  -0.742   2.143   1.00 20.00      O+0
ATOM        7  OXT BTC      0     -1.174   0.443   2.275   1.00 20.00      O+0
ATOM        8  H   BTC      0      1.693   0.682  -1.065   1.00 20.00      H+0
ATOM        9  H2  BTC      0      1.928  -0.454   0.063   1.00 20.00      H+0
ATOM       10  HA  BTC      0     -0.277   1.446   0.042   1.00 20.00      H+0
ATOM       11  HB2 BTC      0     -0.114  -1.526  -0.630   1.00 20.00      H+0
ATOM       12  HB3 BTC      0     -1.604  -0.554  -0.575   1.00 20.00      H+0
ATOM       13  HG  BTC      0     -0.904  -0.965  -3.145   1.00 20.00      H+0
ATOM       14  HXT BTC      0     -1.326   0.158   3.186   1.00 20.00      H+0
CONNECT     1    2    8    9
CONNECT     2    5    3   10    1
CONNECT     3    2   11   12    4
CONNECT     4    3   13
CONNECT     5    2    6    7
CONNECT     6    5
CONNECT     7    5   14
CONNECT     8    1
CONNECT     9    1
CONNECT    10    2
CONNECT    11    3

```

# Drawing small molecules from PDB files

BTC (cysteine) is a small molecule, with only 1 model, 1 “chain”, 1 “residue”

chain = model[' '], since there is an empty field for chain code in the file BTC.pdb

```
parser=PDBParser()
structure=parser.get_structure('cysteine', 'BTC.pdb')
myprint("structure")

model = structure[0]
myprint("model")

chain = model[' ']
myprint("chain")

residue = chain[0]
myprint("residue")
```

```
structure -> <Structure id=cysteine>
```

```
model -> <Model id=0>
```

```
chain -> <Chain id= >
```

```
residue -> <Residue BTC het= resseq=0 icode= >
```



# Drawing small molecules from PDB files

Get the ordered list of atom names in structure

```
for atom in residue:
    print atom.get_serial_number(), atom.get_coord()

myprint("[atom.get_name() for atom in residue]")
```

equivalently:

```
myprint("[atom.get_name() for atom in structure[0][' ')[0]]")
```

```
[atom.get_name() for atom in residue] -> ['N', 'CA', 'CB', 'SG', 'C', 'O', 'OXT',
', 'H', 'H2', 'HA', 'HB2', 'HB3', 'HG', 'HXT']
```

# Drawing small molecules from PDB files

CONECT records are not parsed by Bio.PDB module — we go doing :o)

```
def getPdbConnect (filename):
    myfile = open(filename, 'r')
    for record in myfile:
        terms = record.split()
        if terms[0] == "CONECT": print terms
    myfile.close()
```

```
## units = Angstrom: 1 x 10-10)
```

```
myprint("getPdbConnect('BTC.pdb')")
```

```
getPdbConnect('BTC.pdb') -> ['CONECT', '1', '2', '8', '9']
['CONECT', '2', '5', '3', '10', '1']
['CONECT', '3', '2', '11', '12', '4']
['CONECT', '4', '3', '13']
['CONECT', '5', '2', '6', '7']
['CONECT', '6', '5']
['CONECT', '7', '5', '14']
['CONECT', '8', '1']
['CONECT', '9', '1']
['CONECT', '10', '2']
['CONECT', '11', '3']
['CONECT', '12', '3']
['CONECT', '13', '4']
['CONECT', '14', '7']
None
```

# Drawing small molecules from PDB files

## Select the useful information

```
from pyplasm import *
def getPdbConnect (filename):
    myfile = open(filename, 'r')
    for record in myfile:
        terms = record.split()
        if terms[0] == "CONNECT":
            terms = AA(eval)(terms[1:])
            print terms
    myfile.close()

myprint("getPdbConnect('BTC.pdb')")
```

```
getPdbConnect('BTC.pdb') -> [1, 2, 8, 9]
[2, 5, 3, 10, 1]
[3, 2, 11, 12, 4]
[4, 3, 13]
[5, 2, 6, 7]
[6, 5]
[7, 5, 14]
[8, 1]
[9, 1]
[10, 2]
[11, 3]
[12, 3]
[13, 4]
[14, 7]
None
```

# Drawing small molecules from PDB files

Compute the list of adjacent nodes (terms) to each node

```
from pyplasm import *
def getPdbConnect (filename):
    myfile = open(filename, 'r')
    for record in myfile:
        terms = record.split()
        if terms[0] == "CONNECT":
            node, terms = eval(terms[1]), AA(eval)(terms[2:])
            print node, terms
    myfile.close()

myprint("getPdbConnect('BTC.pdb')")
```

```
getPdbConnect('BTC.pdb') -> 1 [2, 8, 9]
2 [5, 3, 10, 1]
3 [2, 11, 12, 4]
4 [3, 13]
5 [2, 6, 7]
6 [5]
7 [5, 14]
8 [1]
9 [1]
10 [2]
11 [3]
12 [3]
13 [4]
14 [7]
None
```

# Drawing small molecules from PDB files

Compute the **directed** arcs outgoing from each node

```
from pyplasm import *
def getPdbConnect (filename):
    myfile = open(filename, 'r')
    for record in myfile:
        terms = record.split()
        if terms[0] == "CONNECT":
            arcs = DISTL([ eval(terms[1]), AA(eval)(terms[2:]) ])
            print arcs
    myfile.close()

myprint("getPdbConnect('BTC.pdb')")
```

```
getPdbConnect('BTC.pdb') -> [[1, 2], [1, 8], [1, 9]]
[[2, 5], [2, 3], [2, 10], [2, 1]]
[[3, 2], [3, 11], [3, 12], [3, 4]]
[[4, 3], [4, 13]]
[[5, 2], [5, 6], [5, 7]]
[[6, 5]]
[[7, 5], [7, 14]]
[[8, 1]]
[[9, 1]]
[[10, 2]]
[[11, 3]]
[[12, 3]]
[[13, 4]]
```

# Drawing small molecules from PDB files

Compute the **undirected arcs**

```
from pyplasm import *
def getPdbConnect (filename):
    myfile = open(filename, 'r')
    for record in myfile:
        terms = record.split()
        if terms[0] == "CONNECT":
            arcs = DISTL([ eval(terms[1]), AA(eval)(terms[2:]) ])
            arcs = [arc for arc in arcs if arc[0] < arc[1]]
            print arcs
    myfile.close()

myprint("getPdbConnect('BTC.pdb')")
```

```
getPdbConnect('BTC.pdb') -> [[1, 2], [1, 8], [1, 9]]
[[2, 5], [2, 3], [2, 10]]
[[3, 11], [3, 12], [3, 4]]
[[4, 13]]
[[5, 6], [5, 7]]
[]
[[7, 14]]
[]
[]
[]
[]
[]
[]
```

# Drawing small molecules from PDB files

Finally returns the list of undirected arcs

```
from pyplasm import *
def getPDBconnect (filename):
    myfile = open(filename, 'r')
    arcs = []
    for record in myfile:
        terms = record.split()
        if terms[0] == "CONNECT":
            pairs = DISTL([ eval(terms[1]), AA(eval)(terms[2:]) ])
            arcs += [arc for arc in pairs if arc[0] < arc[1]]
    myfile.close()
    return arcs

myprint("getPDBconnect('BTC.pdb')")
```

```
getPDBconnect('BTC.pdb') -> [[1, 2], [1, 8], [1, 9], [2, 5], [2, 3], [2, 10], [3, 11], [3, 12], [3, 4], [4, 13], [5, 6], [5, 7], [7, 14]]
```

# Drawing small molecules from PDB files

Extraction of atom coordinates (3D points), as a [list of array](#)

```
myprint("[atom.get_coord() for atom in residue]")
```

```
[atom.get_coord() for atom in residue] -> [array([ 1.58500004,  0.48300001, -0.081
], dtype=float32), array([ 0.141
,  0.44999999,  0.186
], dtype=float32), array([-0.53299999, -0.52999997, -0.77399999], dtype=float32), array([-0.24699999,  0.004
, -2.48399997], dtype=float32), array([-0.095
,  0.006
,  1.60599995], dtype=float32), array([ 0.685
, -0.74199998,  2.14299989], dtype=float32), array([-1.17400002,  0.44299999,  2.2750001 ], dtype=float32), array([ 1.69299996,  0.68199998, -1.06500006], dtype=float32), array([ 1.92799997, -0.454
,  0.063
], dtype=float32), array([-0.27700001,  1.44599998,  0.042
], dtype=float32), array([-0.114
, -1.52600002, -0.63
], dtype=float32), array([-1.60399997, -0.55400002, -0.57499999], dtype=float32), array([-0.90399998, -0.96499997, -3.14499998], dtype=float32), array([-1.32599998,  0.15800001,  3.18600011], dtype=float32)]
```



# Drawing small molecules from PDB files

Extraction of atom coordinates (3D points), as a [list of lists](#)

```
myprint("[atom.get_coord().tolist() for atom in residue]")
```

```
[atom.get_coord().tolist() for atom in residue] -> [[1.5850000381469727, 0.483000102519989, -0.081000000238418579], [0.14100000262260437, 0.44999998807907104, 0.18600000441074371], [-0.53299999237060547, -0.52999997138977051, -0.77399998903274536], [-0.24699999392032623, 0.0040000001899898052, -2.4839999675750732], [-0.094999998807907104, 0.0060000000521540642, 1.6059999465942383], [0.6850000238418579, -0.74199998378753662, 2.1429998874664307], [-1.1740000247955322, 0.44299998879432678, 2.2750000953674316], [1.6929999589920044, 0.68199998140335083, -1.065000057220459], [1.9279999732971191, -0.45399999618530273, 0.063000001013278961], [-0.27700001001358032, 1.4459999799728394, 0.041999999433755875], [-0.11400000005960464, -1.5260000228881836, -0.62999999523162842], [-1.6039999723434448, -0.55400002002716064, -0.57499998807907104], [-0.90399998426437378, -0.9649999737739563, -3.1449999809265137], [-1.3259999752044678, 0.1580000072717666, 3.1860001087188721]]
```

# Drawing small molecules from PDB files

## Prepare the graph data

```
def graph (filename):
    parser=PDBParser()
    structure=parser.get_structure('molecule', filename)
    model = structure[0]
    chain = model['']
    residue = chain[0]

    nodes = [atom.get_coord().tolist() for atom in residue]
    edges = getPDBconnect('BTC.pdb')
    return nodes,edges

myprint("graph('BTC.pdb')")
```

```
graph('BTC.pdb') -> ([[1.5850000381469727, 0.4830000102519989, -0.0810000002384
18579], [0.14100000262260437, 0.44999998807907104, 0.18600000441074371], [-0.53
299999237060547, -0.52999997138977051, -0.77399998903274536], [-0.2469999939203
2623, 0.0040000001899898052, -2.4839999675750732], [-0.094999998807907104, 0.00
60000000521540642, 1.6059999465942383], [0.68500000238418579, -0.74199998378753
662, 2.1429998874664307], [-1.1740000247955322, 0.44299998879432678, 2.27500009
53674316], [1.6929999589920044, 0.68199998140335083, -1.065000057220459], [1.92
79999732971191, -0.45399999618530273, 0.0630000001013278961], [-0.27700001001358
032, 1.4459999799728394, 0.041999999433755875], [-0.11400000005960464, -1.52600
00228881836, -0.62999999523162842], [-1.6039999723434448, -0.55400002002716064,
-0.57499998807907104], [-0.90399998426437378, -0.9649999737739563, -3.144999980
9265137], [-1.3259999752044678, 0.15800000727176666, 3.1860001087188721]], [[1,
2], [1, 8], [1, 9], [2, 5], [2, 3], [2, 10], [3, 11], [3, 12], [3, 4], [4, 13],
[5, 6], [5, 7], [7, 14]])
```

# Drawing small molecules from PDB files

Return a **geometric value** — i.e. a `<pyplasm.xge.xgepy.Hpc>` object

```
def graph (filename):
    parser=PDBParser()
    structure=parser.get_structure('molecule', filename)
    model = structure[0]
    chain = model[' ']
    residue = chain[0]

    nodes = [atom.get_coord().tolist() for atom in residue]
    edges = getPDBconnect('BTC.pdb')
    return MKPOL([nodes,edges,None])

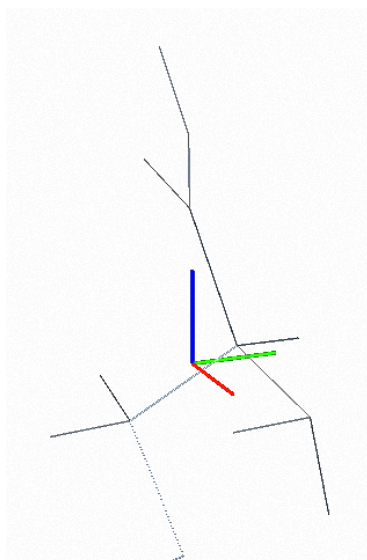
myprint("graph('BTC.pdb')")
```

```
graph('BTC.pdb') -> <pyplasm.xge.xgepy.Hpc; proxy of <Swig Object of type 'std:
:tr1::shared_ptr< Hpc > *' at 0x326c50> >
```

# Drawing small molecules from PDB files

View the 1-complex — embedded in 3D — of the molecule from BTC.pdb

```
VIEW(graph('BTC.pdb'))
```



# Drawing small molecules from PDB files

Get the atomic radiuses from `atomic_radius.py`

```
## http://en.wikipedia.org/wiki/Atomic_radius#Calculated_atomic_radii
## http://en.wikipedia.org/wiki/Atomic_radius#Empirically_measured_atomic_radius
## http://en.wikipedia.org/wiki/Van_der_Waals_radius
## http://en.wikipedia.org/wiki/Covalent_radius

## 0 => No data available
## units = picometers: 1.0 x 10-12

RADIUS_TYPE = 3 # van der Waals

## symbol:(name, empirical, Calculated, van der Waals, covalent)
atomic_radius = {
    "H": ("hydrogen", 35, 53, 120, 38),
    "He": ("helium", 0, 31, 140, 32),
    "Li": ("lithium", 145, 167, 182, 134),
    "Be": ("beryllium", 105, 112, 153, 90),
    "B": ("boron", 85, 87, 192, 82),
    "C": ("carbon", 70, 67, 170, 77),
    "N": ("nitrogen", 65, 56, 155, 75),
    "O": ("oxygen", 60, 48, 152, 73),
    "F": ("fluorine", 50, 42, 147, 71),
    "Ne": ("neon", 0, 38, 154, 69),
    "Na": ("sodium", 180, 190, 227, 154),
    "Mg": ("magnesium", 150, 145, 173, 130),
    "Al": ("aluminium", 125, 118, 184, 118),
    "Si": ("silicon", 110, 111, 210, 111),
    "P": ("phosphorus", 100, 98, 180, 106),
    "S": ("sulfur", 100, 88, 180, 102),
    "Cl": ("chlorine", 100, 79, 175, 99),
```

# Drawing small molecules from PDB files

## Extract atom data (radiuses and type)

```
from atomic_radius import *
```

```
myprint("atomic_radius['Mg']")
myprint("atomic_radius['O'][0:2]")
```

```
myprint("[atom.get_id() for atom in residue]")
myprint("[atom.get_id()[0] for atom in residue]")
myprint("set([atom.get_id()[0] for atom in residue])")
myprint("list(set([atom.get_id()[0] for atom in residue]))")
```

```
atomic_radius['Mg'] -> ('magnesium', 150, 145, 173, 130)
```

```
atomic_radius['O'][0:2] -> ('oxygen', 60)
```

```
[atom.get_id() for atom in residue] -> ['N', 'CA', 'CB', 'SG', 'C', 'O', 'OXT',
'H', 'H2', 'HA', 'HB2', 'HB3', 'HG', 'HXT']
```

```
[atom.get_id()[0] for atom in residue] -> ['N', 'C', 'C', 'S', 'C', 'O', 'O', 'H',
'H', 'H', 'H', 'H', 'H', 'H', 'H']
```

```
set([atom.get_id()[0] for atom in residue]) -> set(['H', 'C', 'S', 'O', 'N'])
```

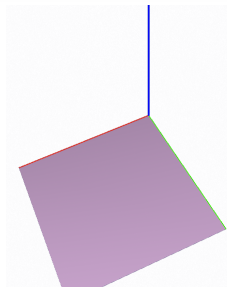
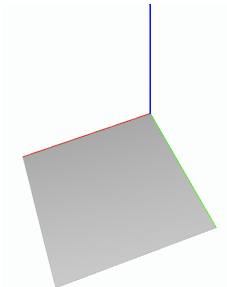
```
list(set([atom.get_id()[0] for atom in residue])) -> ['H', 'C', 'S', 'O', 'N']
```

# Drawing small molecules from PDB files

Atom color definition, according to practice (for biomolecules)

```
atom_color = {
    'H': Color4f([0.8, 0.8, 0.8, 1.0]), # lighth gray
    'C': Color4f([0.3, 0.3, 0.3, 1.0]), # dark gray (quite black)
    'N': BLUE,
    'O': RED,
    'F': Color4f([0.0, 0.75, 1.0, 1.0]), # light blue
    'P': ORANGE,
    'S': YELLOW,
    'Cl': GREEN,
    'K': Color4f([200./255, 162./255, 200./255, 1.0]) # lilac
}

myprint("VIEW(COLOR(atom_color['H'])(CUBOID([1,1])))")
myprint("VIEW(COLOR(atom_color['K'])(CUBOID([1,1])))")
```



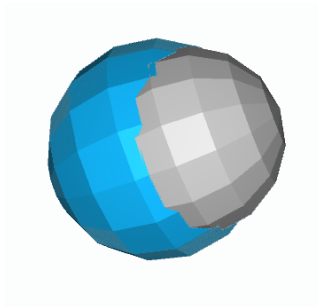
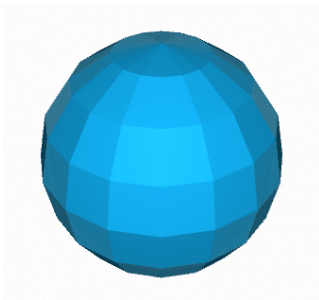
# Drawing small molecules from PDB files

Drawing colored spheres — notice the conversion: picometers → Angstrom

```
def sphere(atom_code):
    return COLOR(atom_color[atom_code])(
        SPHERE(atomic_radius[atom_code][RADIUS_TYPE]/100.)([8,16]))

myprint("VIEW(sphere('F'))")

VIEW(STRUCT([ sphere('F'), T([1,2,3])([0.,.3,.5]), sphere('H') ]))
```





# Drawing small molecules from PDB files

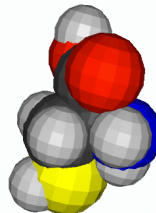
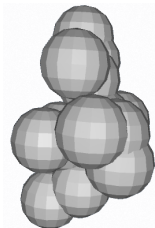
aaaaaaaa

```
def graph2 (filename):
    parser=PDBParser()
    structure=parser.get_structure('molecule', filename)
    model = structure[0]
    chain = model[' ']
    residue = chain[0]

    nodes = [atom.get_coord().tolist() for atom in residue]
    transls = AA(T([1,2,3]))(nodes)
    return STRUCT(CONS(transls)(sphere('H'))))

myprint("graph2('BTC.pdb')")

VIEW(STRUCT([ graph('BTC.pdb'), graph2('BTC.pdb') ]))
```



# Drawing small molecules from PDB files

aaaaaaaa

```
atom_types = [atom.get_id()[0] for atom in residue]
myprint("atom_types")

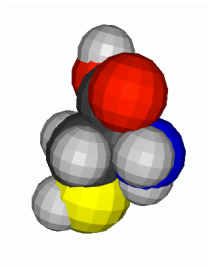
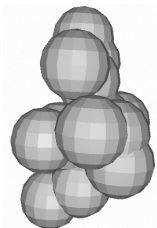
AA(sphere)(atom_types)
```

# Drawing small molecules from PDB files

aaaaaaaa

```
def graph3 (filename):
    parser = PDBParser()
    structure = parser.get_structure('molecule', filename)
    residue = structure[0][' '][0]
    nodes = [atom.get_coord().tolist() for atom in residue]
    transls = AA(T([1,2,3]))(nodes)
    atom_types = [atom.get_id()[0] for atom in residue]
    atoms = AA(sphere)(atom_types)
    return STRUCT(AA(STRUCT)(TRANS([transls,atoms])))

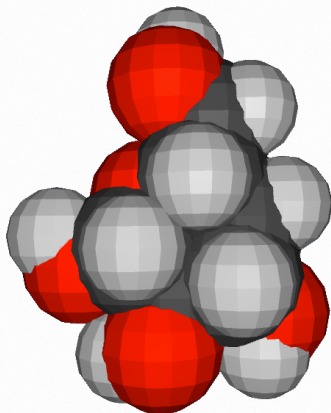
VIEW(graph3('BTC.pdb'))
VIEW(STRUCT([ graph('BTC.pdb'), graph3('BTC.pdb') ]))
```



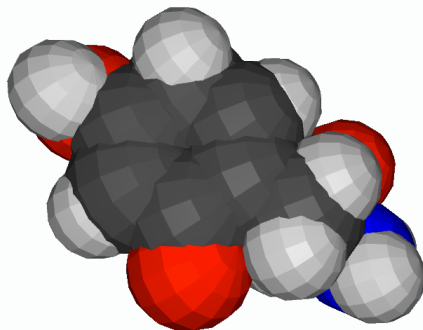
# Drawing small molecules from PDB files

## Other examples

BETA-D-GALACTOSE



2-AMINO-3-(4-HYDROXY-6-  
OXOCYCLOHEXA-1,4-  
DIENYL)PROPANOIC  
ACID



# Viewing PDB-CCD files

- 1 Download and install **pyplasm** from <https://github.com/plasm-language/pyplasm>
- 2 Download the **ccd-viewer** source files
- 3 `$ cd <path>/ccd-viewer`
- 4 `$ python ccd-viewer`

ENJOY !!