OpenBabel et Jmol

OpenBabel

OpenBabel est un ensemble de programme permettant de manipuler et convertir les fichiers de description de molécules dans différents formats.

- Site officiel : http://openbabel.org/wiki/Main_Page
- Interfaçage en Python : http://openbabel.org/wiki/Python

Pour utiliser OpenBabel en python, il faut installer au préalable ces outils. Sous Linux (Debian, Ubuntu,...), installer les paquets openbabel et python-openbabel. Sous Windows, voir cette page, et celle-ci (lien avec python).

Jmol

Jmol est un logiciel libre de visualisation de structures chimiques en 3D, écrit en Java et donc multi-plateformes (Windows, Mac OS X, Linux,...).

C'est un programme idéal pour visualiser des molécules dont les fichiers de description ont été obtenus par OpenBabel (ou d'autres logiciels de chimie).

- Site officiel : http://jmol.sourceforge.net/index.fr.html

Exemple de programme Python

<sxh python; title : generate_alcohols-01.py> #! /usr/bin/env python # -*- coding: utf-8 -*- # génération à partir du code smile d'alcools ddes caractéristiques 3D # et de fichiers .pdb lisible par Jmol # références : # http://openbabel.org/wiki/Python # http://openbabel.org/docs/current/UseTheLibrary/PythonDoc.html#using-iterators # http://pythonchem.blogspot.be/2012/08/fun-and-little-disappointment-with.html # https://gist.github.com/cstein/3294891 # http://nullege.com/codes/search/openbabel

import openbabel

def OBMolMinimize(mol):

```
    """Minimize a molecule
    """
    ff = openbabel.OBForceField.FindForceField("MMFF94")
    ff.Setup(mol)
    ff.ConjugateGradients(100, 1.0e-5)
    ff.GetCoordinates(mol)
    return mol
```

def OBStructureFromSmiles(smilesstring, filename=None):

```
  mol = openbabel.OBMol()
  obConversion = openbabel.OBConversion()
  obConversion.SetInAndOutFormats("smi", "pdb")
  obConversion.ReadString(mol, smilesstring)
  mol.AddHydrogens()
  builder = openbabel.OBBuilder()
  builder.Build(mol)
  mol = OBMolMinimize(mol)
  if filename is None: return mol
  # save structures in subfolder molecules
  obConversion.WriteFile(mol, "molecules/%s.pdb" % filename)
```

def getAlcohols():

```
  molecules = dict()
  molecules['methanol'] = 'CO'
  molecules['ethanol'] = 'CCO'
  molecules['1-propanol'] = 'CCCO'
  molecules['isopropanol'] = 'CC(O)C'
  molecules['n-butanol'] = 'CCCCO'
  molecules['butan-2-ol'] = 'CC(O)CC'
  molecules['isobutanol'] = 'CC(C)CO'
  molecules['tert-butanol'] = 'CC(C)(C)O'
  molecules['1-Pentanol'] = 'CCCCCO'
  return molecules
```

if name == 'main':

```
  etab = openbabel.OBElementTable()
  dico = getAlcohols()
  for elem in dico:
      OBStructureFromSmiles(dico[elem], elem)
```

```
  # Determine the charge of a molecule
  for file_in in dico:
      obConversion = openbabel.OBConversion()
      obConversion.SetInFormat("pdb")
      mol = openbabel.OBMol()
      obConversion.ReadFile(mol, "molecules/%s.pdb" % file_in)
      # get the charges of the atoms
      charge_model = openbabel.OBChargeModel.FindType("MMFF94")
      charge_model.ComputeCharges(mol)
      partial_charges = charge_model.GetPartialCharges()
      print file_in, '- Formula = ', mol.GetFormula()
      print 'mol wt = ', mol.GetMolWt(), '-  Numb atoms = ', mol.NumAtoms(),
'-  Numb bonds = ', mol.NumBonds()
      for atom in openbabel.OBMolAtomIter(mol):
```

```
          z=atom.GetAtomicNum()
          v3=atom.GetVector()
          print atom.GetAtomicMass(), z, etab.GetSymbol(z), 'x=',v3.GetX(),
'y=',v3.GetY(), 'z=',v3.GetZ()
      print 'partial charges = ', partial_charges
      print 'total charge = ', "%i" % int(sum(partial_charges))
      print '********'
```

</sxh> Pour le méthanol, ce programme sort ces informations :

```
methanol - Formula =  CH4O
mol wt =  32.04186 -  Numb atoms =  6 -  Numb bonds =  5
12.0107 6 C x= 0.956 y= -0.086 z= -0.056
15.9994 8 O x= 0.488 y= -1.374 z= 0.299
1.00794 1 H x= 0.587 y= 0.64 z= 0.672
1.00794 1 H x= 0.584 y= 0.177 z= -1.05
1.00794 1 H x= 2.049 y= -0.08 z= -0.052
1.00794 1 H x= 0.831 y= -1.996 z= -0.365
partial charges =  (0.28, -0.68, 0.0, 0.0, 0.0, 0.4)
total charge =  0
```

References

- Open Babel: An open chemical toolbox, Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch and Geoffrey R Hutchison, Journal of Cheminformatics 2011, 3:33 doi:10.1186/1758-2946-3-33
- InChI, the IUPAC International Chemical Identifier Stephen R Heller, Alan McNaught, Igor Pletnev, Stephen Stein and Dmitrii Tchekhovskoi, Journal of Cheminformatics 2015, 7:23 doi:10.1186/s13321-015-0068-4