User Guide for PyChem 1.0

Dongsheng Cao

Yizeng Liang

Table of Contents

1. What is this?

This document is intended to provide an overview of how one can use the PyChem functionality from Python. It's not comprehensive and it's not a manual.
If you find mistakes, or have suggestions for improvements, please either fix them yourselves in the source document (the .py file) or send them to the mailing list: oriental-cds@hotmail.com

2. Install the PyChem package

PyChem has been successfully tested on Linux and Windows systems. The author could download the PyChem package via: http://code.google.com/p/pychem/downloads/list (.zip and .tar.gz). The install process of PyChem is very easy:
****
* You first need to install RDKit, Openbabel, MOPAC and pybel successfully.*
****
Openbabel and pybel can be downloaded via: http://openbabel.org/wiki/Main_Page
RDkit can be downloaded via: http://code.google.com/p/rdkit/
MOPAC can be downloaded via: http://openmopac.net/
Note: PyChem was tested in MOPAC 7.

On Windows:

(1): download the pychem package (.zip)

(2): extract or uncompress the .zip file

(3): cd pychem-1.0

(4): python setup.py install

On Linux:

(1): download the pychem package (.tar.gz)

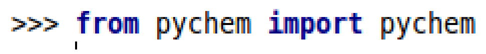
(2): tar -zxf pychem-1.0.tar.gz

(3): cd pychem-1.0

(4): python setup.py install or sudo python setup.py install

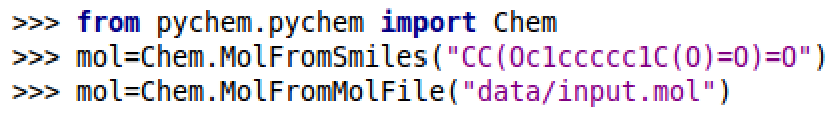
3. Read molecules

The majority of the basic molecular functionality is found in module pychem:



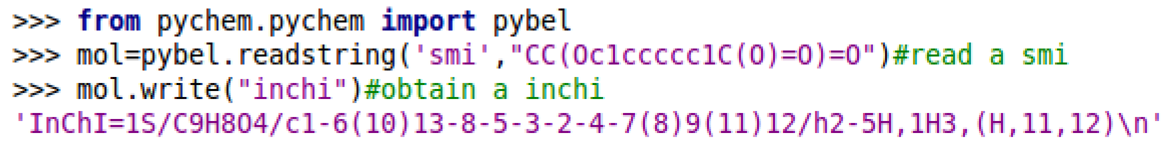
```
>>> from pychem import pychem
```

Individual molecule can be constructed using a variety of approaches.



```
>>> from pychem.pychem import Chem
>>> mol=Chem.MolFromSmiles("CC(Oc1ccccc1C(O)=O)=O")
>>> mol=Chem.MolFromMolFile("data/input.mol")
```

Because we have imported pybel module, all functionalities in pybel can be used to construct a Mol object. Moreover, the transformation between any two molecular formats are allowed by using pybel.

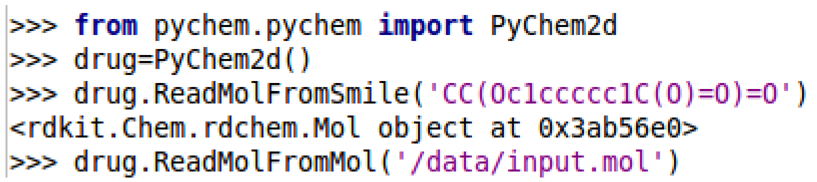


```
>>> from pychem.pychem import pybel
>>> mol=pybel.readstring('smi',"CC(Oc1ccccc1C(O)=O)=O")#read a smi
>>> mol.write("inchi")#obtain a inchi
'InChI=1S/C9H8O4/c1-6(10)13-8-5-3-2-4-7(8)9(11)12/h2-5H,1H3,(H,11,12)\n'
```

Note: When computing 2-D descriptors by individual module, we used Mol object from RDKit. When computing 3-D descriptors by individual module, we used Mol object from pybel. The compatibility between PyChem and other packages ensures that PyChem could be conveniently transplanted.

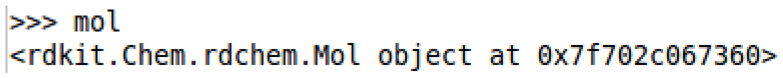
The PyChem allow the users to provide different molecular formats when using PyChem2d object or PyChem3d object.

Using PyChem2d:



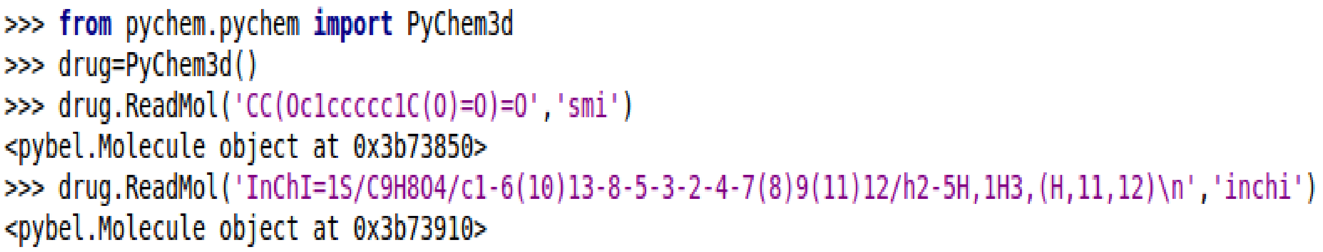
```
>>> from pychem.pychem import PyChem2d
>>> drug=PyChem2d()
>>> drug.ReadMolFromSmile('CC(Oc1ccccc1C(O)=O)=O')
<rdkit.Chem.rdchem.Mol object at 0x3ab56e0>
>>> drug.ReadMolFromMol('/data/input.mol')
```

All of these functions return a Mol object on success:



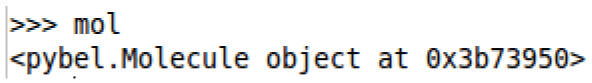
```
>>> mol
<rdkit.Chem.rdchem.Mol object at 0x7f702c067360>
```

Using PyChem3d:



```
>>> from pychem.pychem import PyChem3d
>>> drug=PyChem3d()
>>> drug.ReadMol('CC(Oc1ccccc1C(O)=O)=O','smi')
<pybel.Molecule object at 0x3b73850>
>>> drug.ReadMol('InChI=1S/C9H8O4/c1-6(10)13-8-5-3-2-4-7(8)9(11)12/h2-5H,1H3,(H,11,12)\n','inchi')
<pybel.Molecule object at 0x3b73910>
```

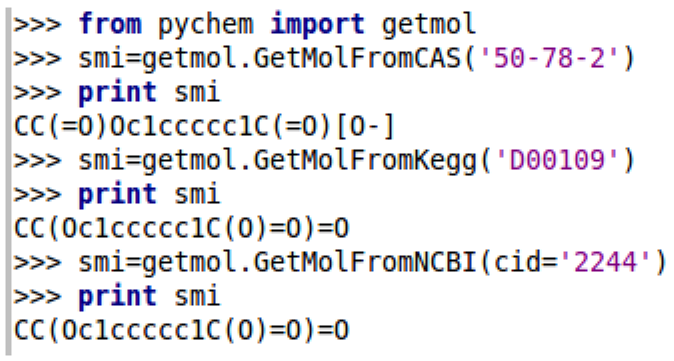
All of these functions return a Mol object on success:



```
>>> mol
<pybel.Molecule object at 0x3b73950>
```

4. Download molecules from corresponding ID

The PyDPI allows the user to download the molecules by providing their IDs such as CAS, NCBI, KEGG, EBI and Drugbank.



```
>>> from pychem import getmol
>>> smi=getmol.GetMolFromCAS('50-78-2')
>>> print smi
CC(=O)Oc1ccccc1C(=O)[O-]
>>> smi=getmol.GetMolFromKegg('D00109')
>>> print smi
CC(Oc1ccccc1C(O)=O)=O
>>> smi=getmol.GetMolFromNCBI(cid='2244')
>>> print smi
CC(Oc1ccccc1C(O)=O)=O
```

By providing a aspirin IDs, we could download its SMILES format conveniently.

We can also download and read a molecule by constructing a PyChem2d or PyChem3d object, which contains the majority of the basic drug molecular functionality.

Using PyChem2d object:

```
>>> drug=pychem.PyChem2d()
>>> smi=drug.GetMolFromNCBI('2244')
>>> print smi
CC(Oc1ccccc1C(O)=O)=O
>>> drug.ReadMolFromSmile(smi)
<rdkit.Chem.rdchem.Mol object at 0x423d980>
```

Using PyChem3d object:

```
>>> drug=pychem.PyChem3d()
>>> smi=drug.GetMolFromNCBI('2244')
>>> print smi
CC(Oc1ccccc1C(O)=O)=O
>>> mol=drug.ReadMol(smi)
>>> print mol
CC(=O)Oc1ccccc1C(=O)O
```

You could read a molecule by providing a KEGG ID:

```
>>> drug=pychem.PyChem2d() #construct a PyChem2d object
>>> smi=drug.GetMolFromKegg('D00109') #download a molecule with id D01276
>>> drug.ReadMolFromSmile(smi) #read a molecule
<rdkit.Chem.rdchem.Mol object at 0x423df30>
```

5. Calculating molecular descriptors

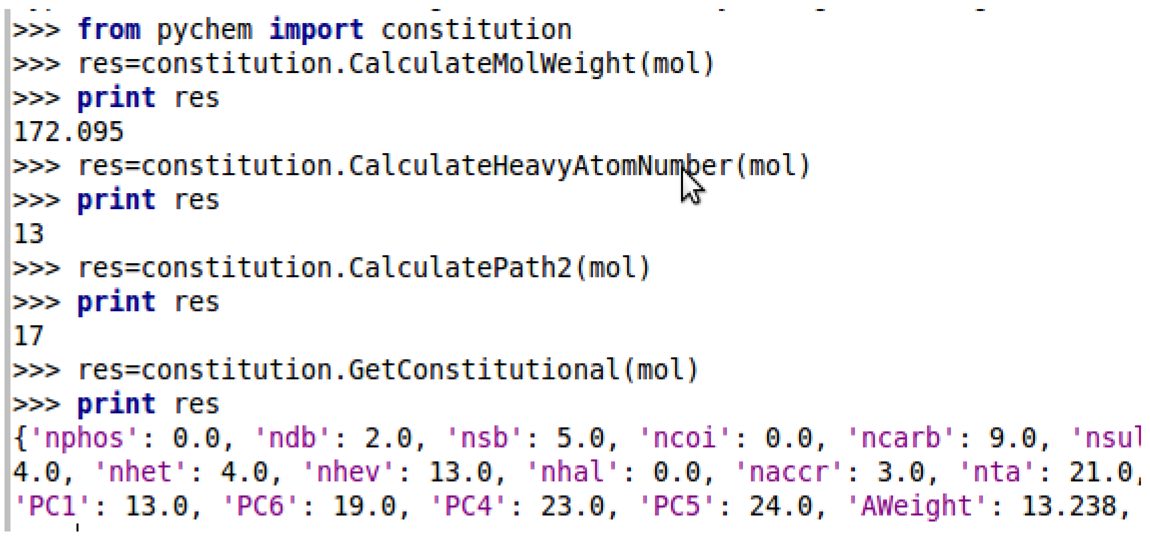
The PyChem package could calculate a large number of molecular descriptors. These descriptors capture and magnify distinct aspects of chemical structures. Generally speaking, all descriptors could be divided into two classes: 2-D descriptors and 3-D descriptors. 2-D descriptors only used the property of molecular topology, including constitutional descriptors, topological descriptors, connectivity indices, E-state indices, Basak information indices, Burden descriptors, autocorrelation descriptors, charge descriptors, molecular properties, kappa shape indices, MOE-type descriptors, and molecular fingerprints. 3-D descriptors need the optimization of molecular structure. In PyChem 1.0, we used a widely used MOPAC program to optimize molecular structures by the AM1 method. The 3-D descriptors calculated by PyChem include geometric descriptors, CPSA descriptors, RDF descriptors, MoRSE descriptors, and WHIM descriptors. The PyChem package could compute 1135 molecular descriptors.

Once we read a Mol object, we could easily calculate these molecular descriptors:

5.1. Calculating 2-D descriptors

We could import the corresponding module to calculate the molecular descriptors as need. There are 13 modules to compute 2-D descriptors. Moreover, a easier way to compute these descriptors is construct a PyChem2d object, which encapsulates all methods for the calculation of 2-D descriptors.

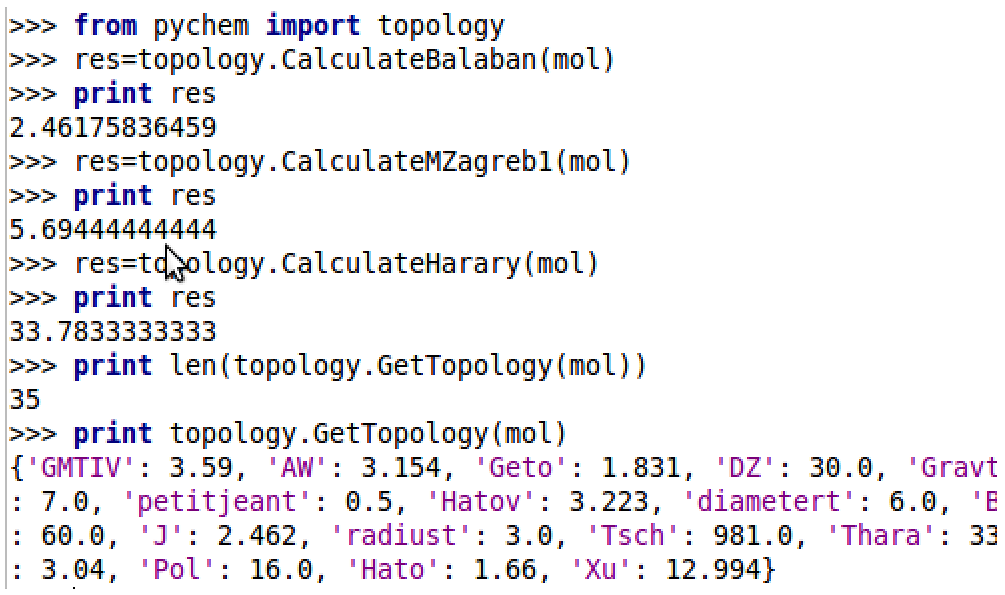
5.1.1. Molecular constitutional descriptors



```
>>> from pychem import constitution
>>> res=constitution.CalculateMolWeight(mol)
>>> print res
172.095
>>> res=constitution.CalculateHeavyAtomNumber(mol)
>>> print res
13
>>> res=constitution.CalculatePath2(mol)
>>> print res
17
>>> res=constitution.GetConstitutional(mol)
>>> print res
{'nphos': 0.0, 'ndb': 2.0, 'nsb': 5.0, 'ncoi': 0.0, 'ncarb': 9.0, 'nsul
4.0, 'nhet': 4.0, 'nhev': 13.0, 'nhal': 0.0, 'naccr': 3.0, 'nta': 21.0,
'PC1': 13.0, 'PC6': 19.0, 'PC4': 23.0, 'PC5': 24.0, 'AWeight': 13.238,
```

We could calculate any constitution descriptor by calling the corresponding functions. We could also calculate all 30 descriptors by calling GetConstitution function. The result is given in the form of dictionary.

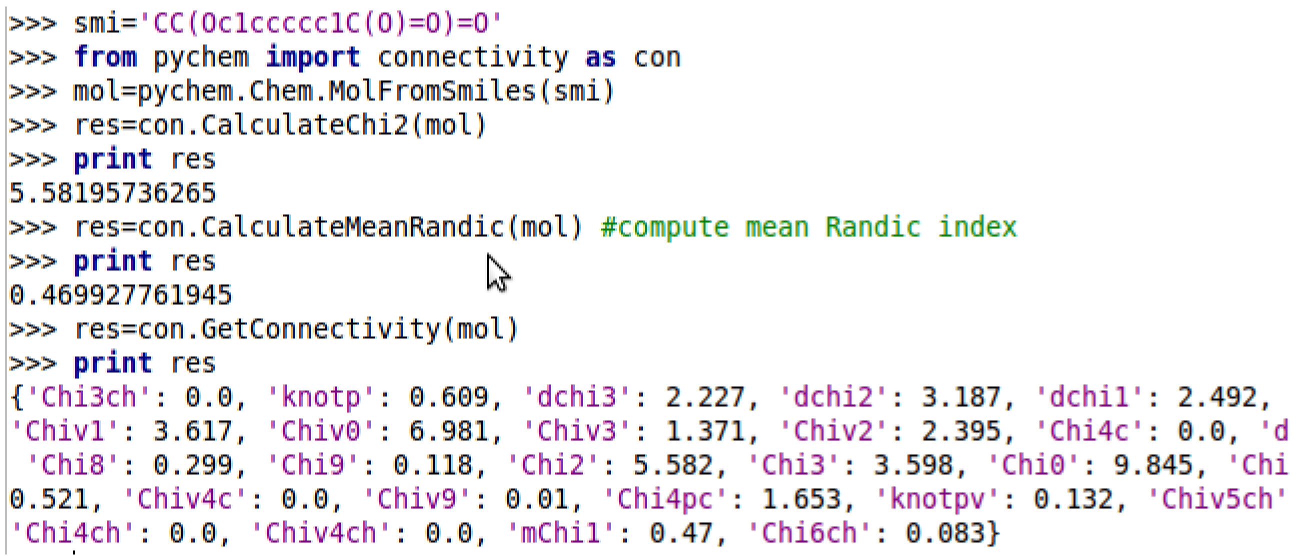
5.1.2. Topology descriptors



```
>>> from pychem import topology
>>> res=topology.CalculateBalaban(mol)
>>> print res
2.46175836459
>>> res=topology.CalculateMZagreb1(mol)
>>> print res
5.69444444444
>>> res=topology.CalculateHarary(mol)
>>> print res
33.7833333333
>>> print len(topology.GetTopology(mol))
35
>>> print topology.GetTopology(mol)
{'GMTIV': 3.59, 'AW': 3.154, 'Geto': 1.831, 'DZ': 30.0, 'Gravt
: 7.0, 'petitjeant': 0.5, 'Hatov': 3.223, 'diametert': 6.0, 'E
: 60.0, 'J': 2.462, 'radiust': 3.0, 'Tsch': 981.0, 'Thara': 33
: 3.04, 'Pol': 16.0, 'Hato': 1.66, 'Xu': 12.994}
```

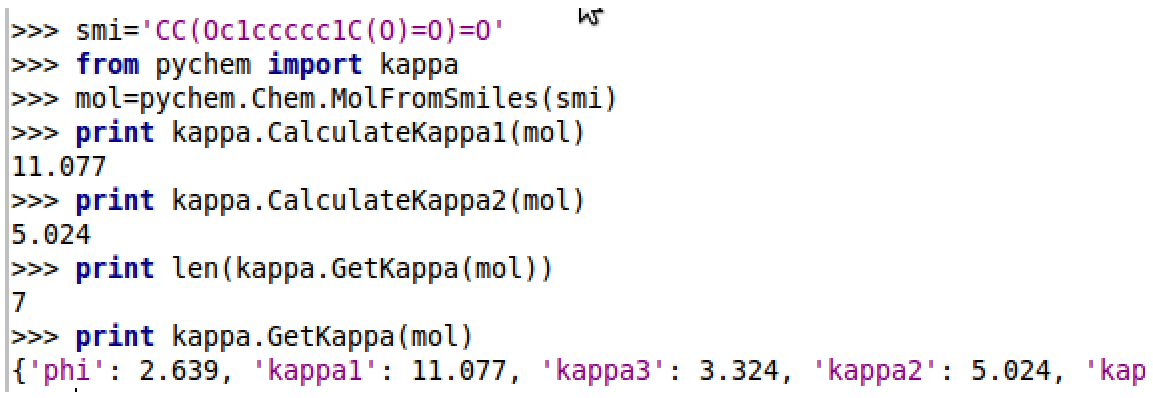
35 topology descriptors can be calculated by the PyChem package. For detailed information of topology descriptors, refer to Table S1 in Appendix and their introductions in Manual.

5.1.3. Molecular connectivity indices



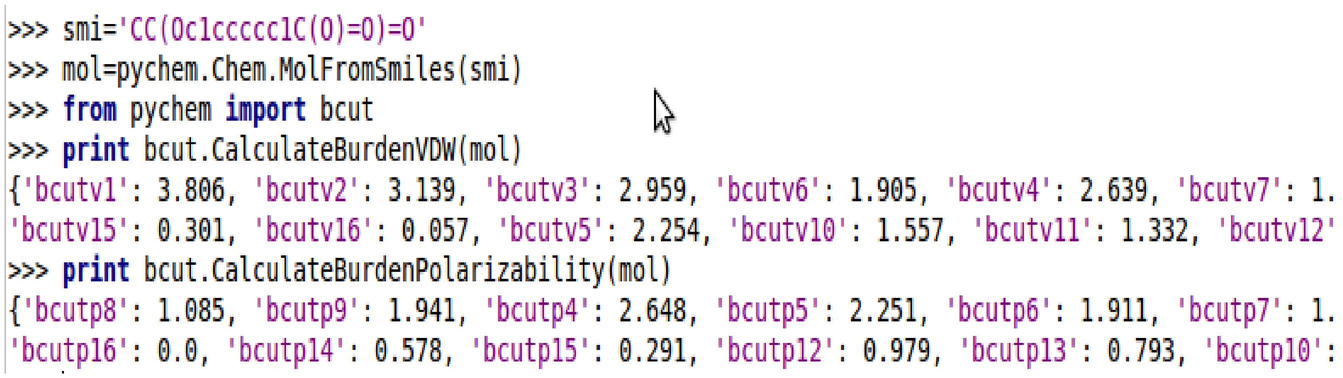
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> from pychem import connectivity as con
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> res=con.CalculateChi2(mol)
>>> print res
5.58195736265
>>> res=con.CalculateMeanRandic(mol) #compute mean Randic index
>>> print res
0.469927761945
>>> res=con.GetConnectivity(mol)
>>> print res
{'Chi3ch': 0.0, 'knotp': 0.609, 'dchi3': 2.227, 'dchi2': 3.187, 'dchi1': 2.492,
'Chiv1': 3.617, 'Chiv0': 6.981, 'Chiv3': 1.371, 'Chiv2': 2.395, 'Chi4c': 0.0, 'd
 'Chi8': 0.299, 'Chi9': 0.118, 'Chi2': 5.582, 'Chi3': 3.598, 'Chi0': 9.845, 'Chi
0.521, 'Chiv4c': 0.0, 'Chiv9': 0.01, 'Chi4pc': 1.653, 'knotpv': 0.132, 'Chiv5ch'
'Chi4ch': 0.0, 'Chiv4ch': 0.0, 'mChi1': 0.47, 'Chi6ch': 0.083}
```

5.1.4. Kappa shape descriptors



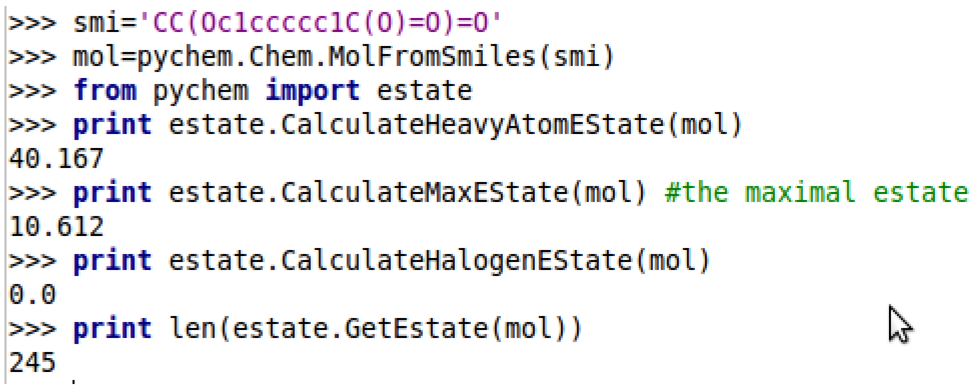
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> from pychem import kappa
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> print kappa.CalculateKappa1(mol)
11.077
>>> print kappa.CalculateKappa2(mol)
5.024
>>> print len(kappa.GetKappa(mol))
7
>>> print kappa.GetKappa(mol)
{'phi': 2.639, 'kappa1': 11.077, 'kappa3': 3.324, 'kappa2': 5.024, 'kap
```

5.1.5. Burden descriptors



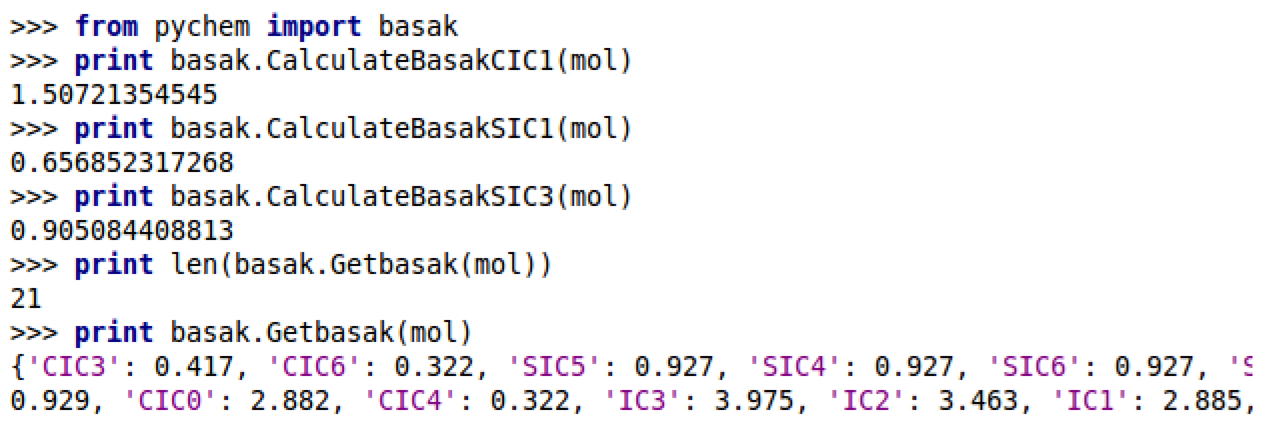
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> from pychem import bcut
>>> print bcut.CalculateBurdenVDW(mol)
{'bcutv1': 3.806, 'bcutv2': 3.139, 'bcutv3': 2.959, 'bcutv6': 1.905, 'bcutv4': 2.639, 'bcutv7': 1.
'bcutv15': 0.301, 'bcutv16': 0.057, 'bcutv5': 2.254, 'bcutv10': 1.557, 'bcutv11': 1.332, 'bcutv12'
>>> print bcut.CalculateBurdenPolarizability(mol)
{'bcutp8': 1.085, 'bcutp9': 1.941, 'bcutp4': 2.648, 'bcutp5': 2.251, 'bcutp6': 1.911, 'bcutp7': 1.
'bcutp16': 0.0, 'bcutp14': 0.578, 'bcutp15': 0.291, 'bcutp12': 0.979, 'bcutp13': 0.793, 'bcutp10':
```

5.1.6. E-state indices



```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> from pychem import estate
>>> print estate.CalculateHeavyAtomEState(mol)
40.167
>>> print estate.CalculateMaxEState(mol) #the maximal estate
10.612
>>> print estate.CalculateHalogenEState(mol)
0.0
>>> print len(estate.GetEstate(mol))
245
```

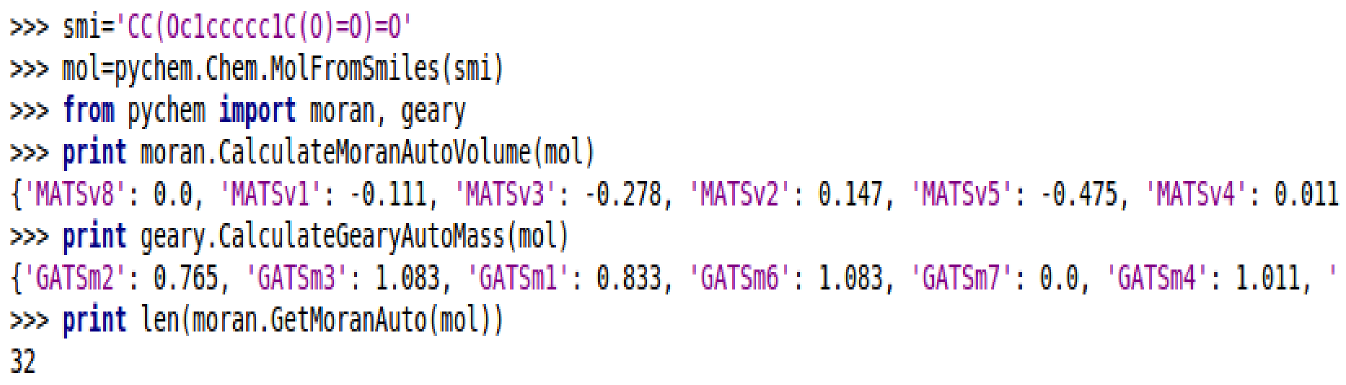
5.1.7. Basak information indices



```
>>> from pychem import basak
>>> print basak.CalculateBasakCIC1(mol)
1.50721354545
>>> print basak.CalculateBasakSIC1(mol)
0.656852317268
>>> print basak.CalculateBasakSIC3(mol)
0.905084408813
>>> print len(basak.Getbasak(mol))
21
>>> print basak.Getbasak(mol)
{'CIC3': 0.417, 'CIC6': 0.322, 'SIC5': 0.927, 'SIC4': 0.927, 'SIC6': 0.927, 'S
0.929, 'CIC0': 2.882, 'CIC4': 0.322, 'IC3': 3.975, 'IC2': 3.463, 'IC1': 2.885,
```

5.1.8. Autocorrelation descriptors

There are three types of autocorrelation descriptors in the PyChem package: Moreau-Broto, Moran, Geary.



```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> from pychem import moran, geary
>>> print moran.CalculateMoranAutoVolume(mol)
{'MATSv8': 0.0, 'MATSv1': -0.111, 'MATSv3': -0.278, 'MATSv2': 0.147, 'MATSv5': -0.475, 'MATSv4': 0.011
>>> print geary.CalculateGearyAutoMass(mol)
{'GATSm2': 0.765, 'GATSm3': 1.083, 'GATSm1': 0.833, 'GATSm6': 1.083, 'GATSm7': 0.0, 'GATSm4': 1.011, '
>>> print len(moran.GetMoranAuto(mol))
32
```

5.1.9. Molecular properties

```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> from pychem import molproperty as mp
>>> print mp.CalculateMolLogP(mol)
1.31
>>> print mp.CalculateMolMR(mol)
44.71
>>> print mp.CalculateTPSA(mol)
63.6
>>> print mp.GetMolecularProperty(mol)
{'TPSA': 63.6, 'Hy': -2.562, 'LogP': 1.31, 'LogP2': 1.716, 'UI': 3.17, 'MR': 44.71}
```

5.1.10. Charge descriptors

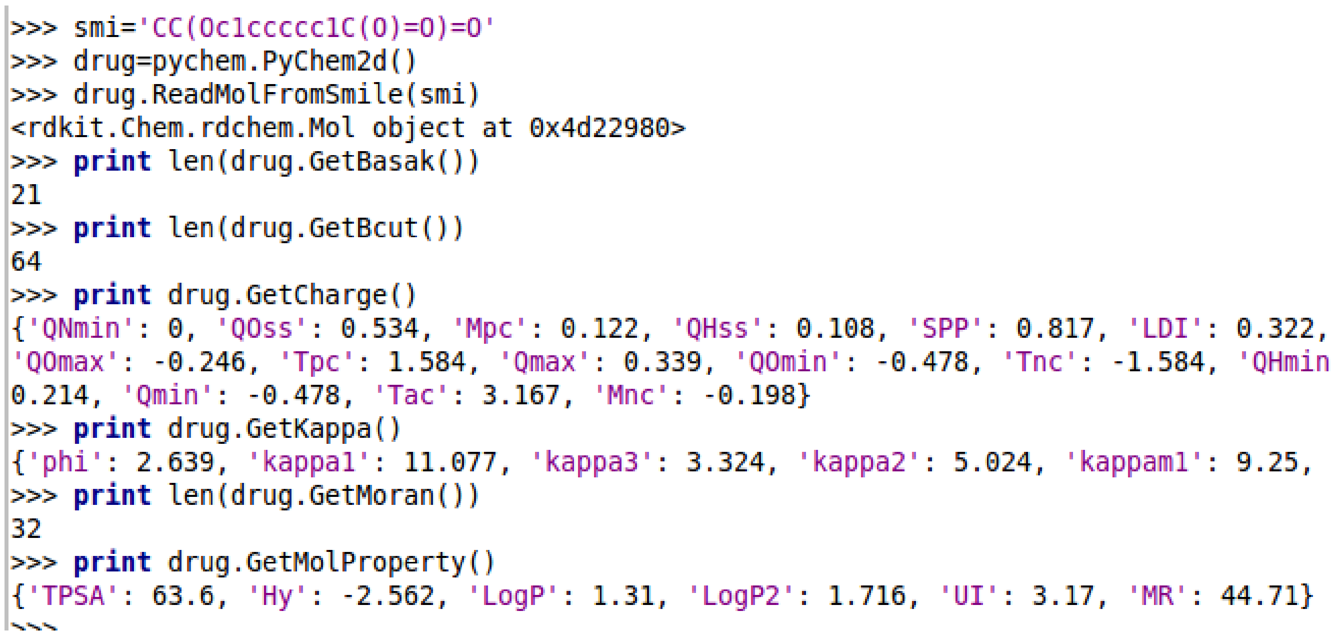
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> from pychem import charge
>>> print charge.CalculateLocalDipoleIndex(mol)
0.322
>>> print charge.CalculateAllSumSquareCharge(mol)
0.893
>>> print len(charge.GetCharge(mol))
25
```

5.1.11. MOE-type descriptors

```
>>> from pychem import moe
>>> print moe.CalculateTPSA(mol)
{'TPSA1': 63.6}
>>> print moe.CalculatePEOEVSA(mol)
{'PEOEVSA11': 0.0, 'PEOEVSA10': 0.0, 'PEOEVSA3': 0.0, 'PEOEVSA2': 4.795, 'PEOE
'PEOEVSA5': 12.133, 'PEOEVSA4': 0.0, 'PEOEVSA13': 11.939, 'PEOEVSA9': 11.313,
>>> print len(moe.GetMOE(mol))
60
```

5.1.12. Using PyChem2d object

A easier way to calculate molecular descriptors is to generate a PyChem2d object and then call their methods. The PyChem2d contains the majority of drug molecule operation functionality.

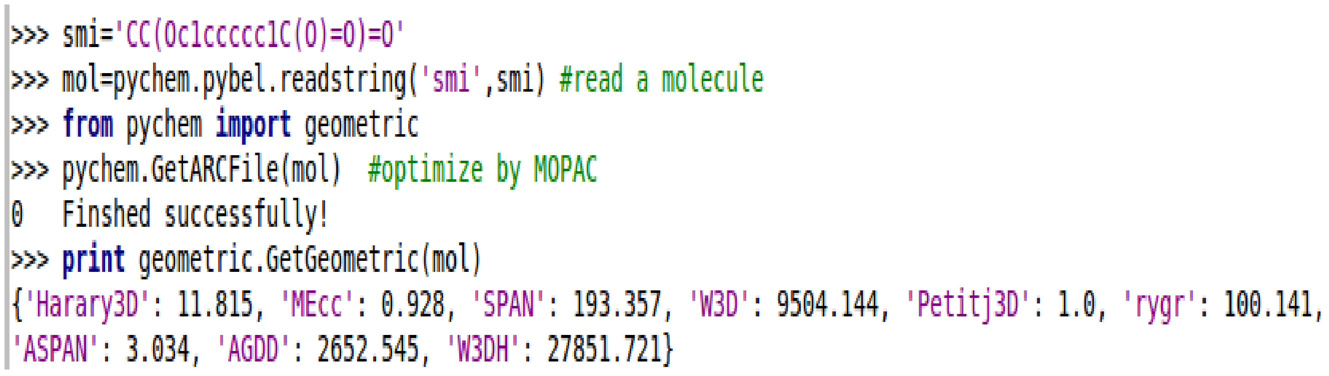


```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> drug=pychem.PyChem2d()
>>> drug.ReadMolFromSmile(smi)
<rdkit.Chem.rdchem.Mol object at 0x4d22980>
>>> print len(drug.GetBasak())
21
>>> print len(drug.GetBcut())
64
>>> print drug.GetCharge()
{'QNmin': 0, 'QOss': 0.534, 'Mpc': 0.122, 'QHss': 0.108, 'SPP': 0.817, 'LDI': 0.322,
'QOmax': -0.246, 'Tpc': 1.584, 'Qmax': 0.339, 'QOmin': -0.478, 'Tnc': -1.584, 'QHmin
0.214, 'Qmin': -0.478, 'Tac': 3.167, 'Mnc': -0.198}
>>> print drug.GetKappa()
{'phi': 2.639, 'kappa1': 11.077, 'kappa3': 3.324, 'kappa2': 5.024, 'kappam1': 9.25,
>>> print len(drug.GetMoran())
32
>>> print drug.GetMolProperty()
{'TPSA': 63.6, 'Hy': -2.562, 'LogP': 1.31, 'LogP2': 1.716, 'UI': 3.17, 'MR': 44.71}
>>>
```

5.2. Calculating 3-D descriptors

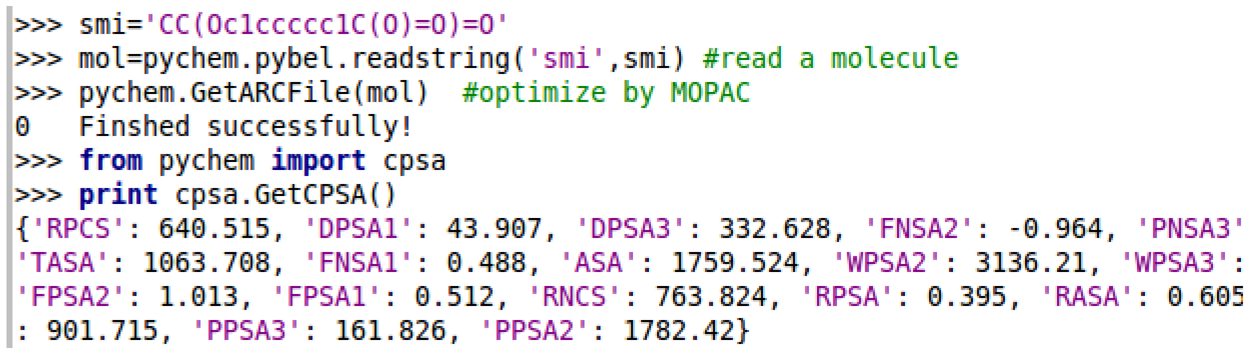
The 3-D molecular descriptors calculated by PyChem include geometric descriptors, CPSA descriptors, RDF descriptors, MoRSE descriptors, and WHIM descriptors. We could import the corresponding module to calculate the molecular descriptors as need. There are 5 modules to compute 3-D descriptors. Moreover, a easier way to compute these descriptors is construct a PyChem3d object, which encapsulates all methods for the calculation of 3-D descriptors.

5.2.1. Geometric descriptors



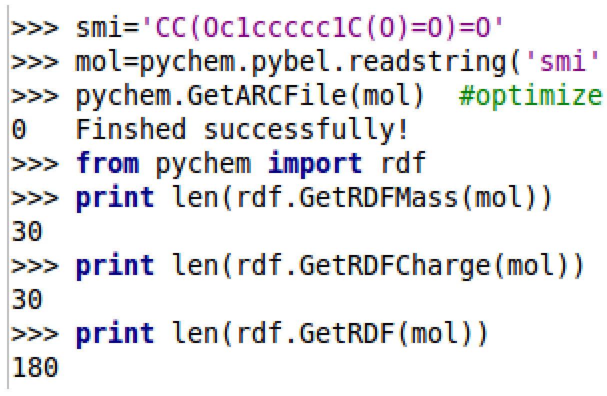
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.pybel.readstring('smi',smi) #read a molecule
>>> from pychem import geometric
>>> pychem.GetARCFile(mol)  #optimize by MOPAC
0   Finshed successfully!
>>> print geometric.GetGeometric(mol)
{'Harary3D': 11.815, 'MEcc': 0.928, 'SPAN': 193.357, 'W3D': 9504.144, 'Petitj3D': 1.0, 'rygr': 100.141,
'ASPAN': 3.034, 'AGDD': 2652.545, 'W3DH': 27851.721}
```

5.2.2. CPSA descriptors



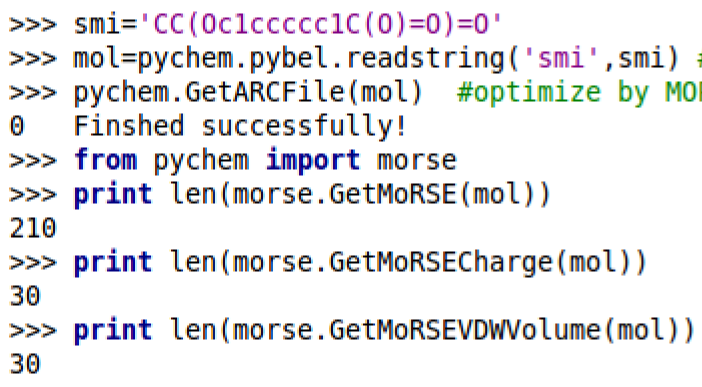
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.pybel.readstring('smi',smi) #read a molecule
>>> pychem.GetARCFile(mol)  #optimize by MOPAC
0   Finshed successfully!
>>> from pychem import cpsa
>>> print cpsa.GetCPSA()
{'RPCS': 640.515, 'DPSA1': 43.907, 'DPSA3': 332.628, 'FNSA2': -0.964, 'PNSA3'
'TASA': 1063.708, 'FNSA1': 0.488, 'ASA': 1759.524, 'WPSA2': 3136.21, 'WPSA3':
'FPSA2': 1.013, 'FPSA1': 0.512, 'RNCS': 763.824, 'RPSA': 0.395, 'RASA': 0.605
: 901.715, 'PPSA3': 161.826, 'PPSA2': 1782.42}
```

5.2.3. RDF descriptors



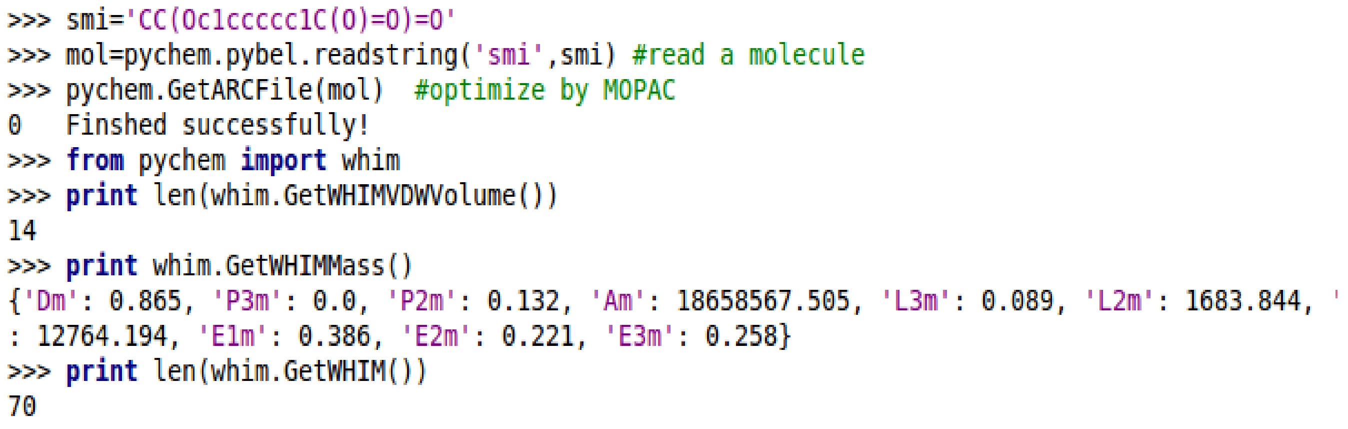
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.pybel.readstring('smi',smi) #read a molecule
>>> pychem.GetARCFile(mol)  #optimize by MOPAC
0   Finshed successfully!
>>> from pychem import rdf
>>> print len(rdf.GetRDFMass(mol))
30
>>> print len(rdf.GetRDFCharge(mol))
30
>>> print len(rdf.GetRDF(mol))
180
```

5.2.4. MoRSE descriptors



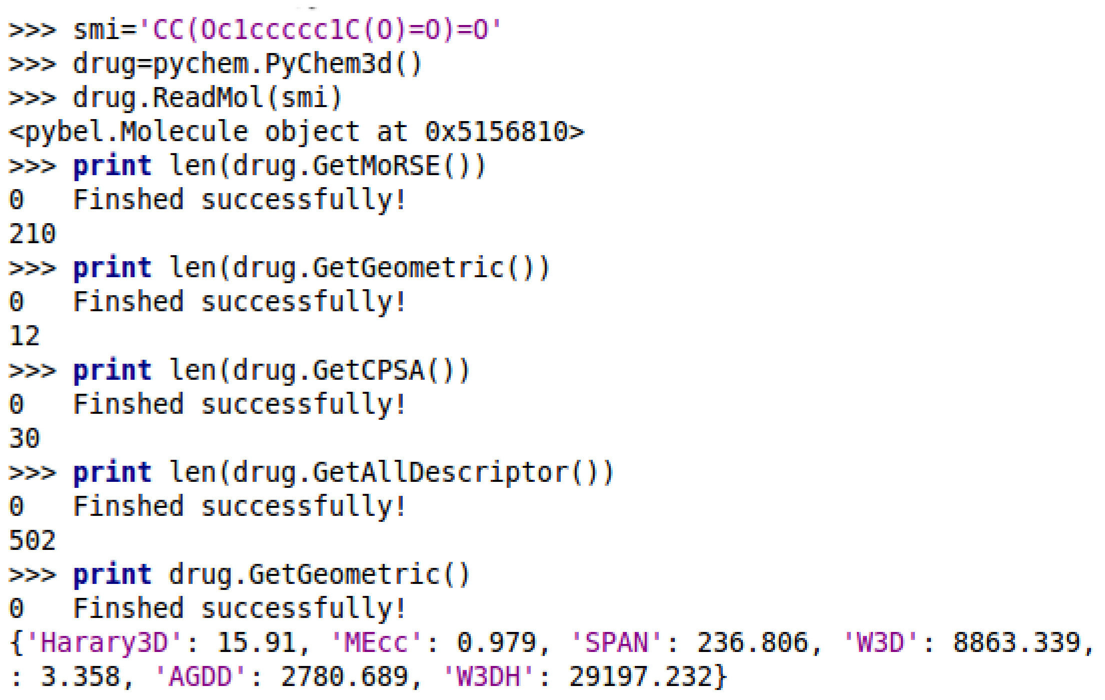
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.pybel.readstring('smi',smi) #read a molecule
>>> pychem.GetARCFile(mol)  #optimize by MOPAC
0   Finshed successfully!
>>> from pychem import morse
>>> print len(morse.GetMoRSE(mol))
210
>>> print len(morse.GetMoRSECharge(mol))
30
>>> print len(morse.GetMoRSEVDWVolume(mol))
30
```

5.2.5. WHIM descriptors



```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> mol=pychem.pybel.readstring('smi',smi) #read a molecule
>>> pychem.GetARCFile(mol)  #optimize by MOPAC
0   Finshed successfully!
>>> from pychem import whim
>>> print len(whim.GetWHIMVDWVolume())
14
>>> print whim.GetWHIMMass()
{'Dm': 0.865, 'P3m': 0.0, 'P2m': 0.132, 'Am': 18658567.505, 'L3m': 0.089, 'L2m': 1683.844, '
: 12764.194, 'E1m': 0.386, 'E2m': 0.221, 'E3m': 0.258}
>>> print len(whim.GetWHIM())
70
```

5.2.6. Using PyChem3d object

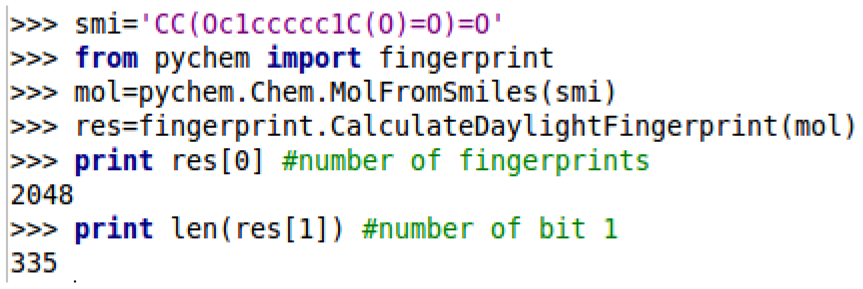


```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> drug=pychem.PyChem3d()
>>> drug.ReadMol(smi)
<pybel.Molecule object at 0x5156810>
>>> print len(drug.GetMoRSE())
0   Finshed successfully!
210
>>> print len(drug.GetGeometric())
0   Finshed successfully!
12
>>> print len(drug.GetCPSA())
0   Finshed successfully!
30
>>> print len(drug.GetAllDescriptor())
0   Finshed successfully!
502
>>> print drug.GetGeometric()
0   Finshed successfully!
{'Harary3D': 15.91, 'MEcc': 0.979, 'SPAN': 236.806, 'W3D': 8863.339,
: 3.358, 'AGDD': 2780.689, 'W3DH': 29197.232}
```

5.3. Molecular fingerprints and chemoinforamtics

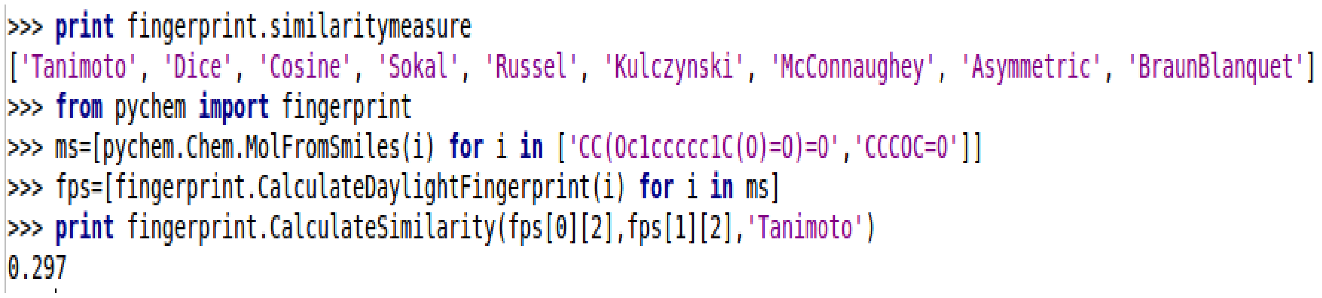
In the PyChem package, there are seven types of molecular fingerprints which are defined by abstracting and magnifying different aspects of molecular topology.

5.3.1. Daylight-type fingerprints



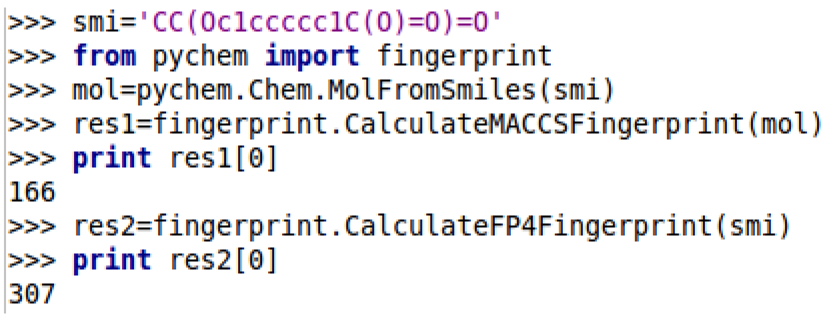
```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> from pychem import fingerprint
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> res=fingerprint.CalculateDaylightFingerprint(mol)
>>> print res[0] #number of fingerprints
2048
>>> print len(res[1]) #number of bit 1
335
```

We can calculate the similarity between two molecules by specifying a type of similarity measure. There exist to be nine types of similarity measures to calculate the similarity between two molecules.



```
>>> print fingerprint.similaritymeasure
['Tanimoto', 'Dice', 'Cosine', 'Sokal', 'Russel', 'Kulczynski', 'McConnaughey', 'Asymmetric', 'BraunBlanquet']
>>> from pychem import fingerprint
>>> ms=[pychem.Chem.MolFromSmiles(i) for i in ['CC(Oc1ccccc1C(O)=O)=O','CCCOC=O']]
>>> fps=[fingerprint.CalculateDaylightFingerprint(i) for i in ms]
>>> print fingerprint.CalculateSimilarity(fps[0][2],fps[1][2],'Tanimoto')
0.297
```

5.3.2. MACCS keys and FP4 fingerprints



```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> from pychem import fingerprint
>>> mol=pychem.Chem.MolFromSmiles(smi)
>>> res1=fingerprint.CalculateMACCSFingerprint(mol)
>>> print res1[0]
166
>>> res2=fingerprint.CalculateFP4Fingerprint(smi)
>>> print res2[0]
307
```

Note that the input of MACCS and FP4 is different.

5.3.3. E-state fingerprints

```
>>> from pychem.pychem import Chem
>>> from pychem import fingerprint
>>> mol=Chem.MolFromSmiles('CC(Oc1ccccc1C(O)=O)=O')
>>> fp=fingerprint.CalculateEstateFingerprint(mol)
>>> print fp[0]
79
>>> print fp[1]
{'12': 1, '17': 1, '16': 1, '36': 1, '35': 1, '34': 1, '7': 1}
```

5.3.4. Atom pairs and topological torsions

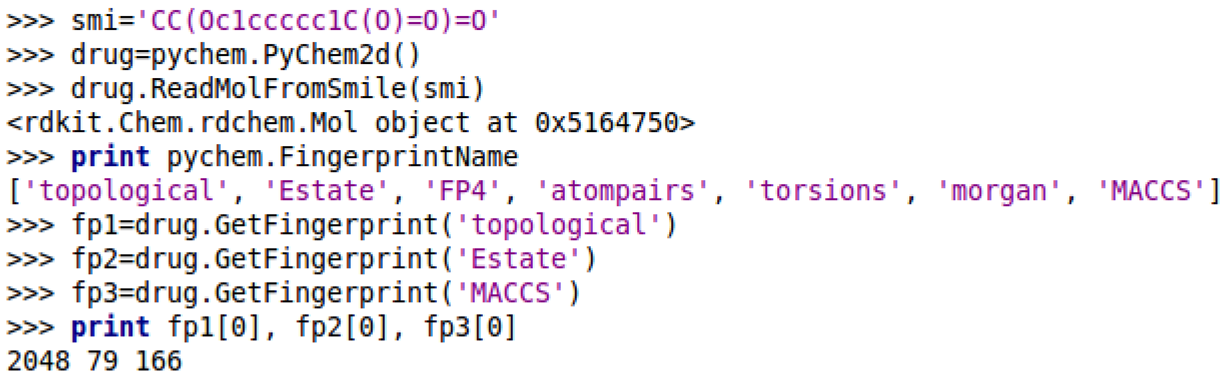
```
>>> from pychem.pychem import Chem
>>> from pychem import fingerprint
>>> mol=Chem.MolFromSmiles('CC(Oc1ccccc1C(O)=O)=O')
>>> fp1=fingerprint.CalculateAtomPairsFingerprint(mol)
>>> fp2=fingerprint.CalculateTopologicalTorsionFingerprint(mol)
>>> print fp1[0], fp2[0]
8388608 68719476735
>>> print len(fp1[1]),len(fp2[1])
51 16
```

5.3.5. Morgan fingerprints

```
>>> from pychem.pychem import Chem
>>> from pychem import fingerprint, getmol
>>> smi=getmol.GetMolFromNCBI('2244')
>>> mol=Chem.MolFromSmiles(smi)
>>> fp=fingerprint.CalculateMorganFingerprint(mol,radius=2)
>>> print fp[0]
4294967295
>>> print len(fp[1])
25
```

5.3.6. Using PyChem2d object

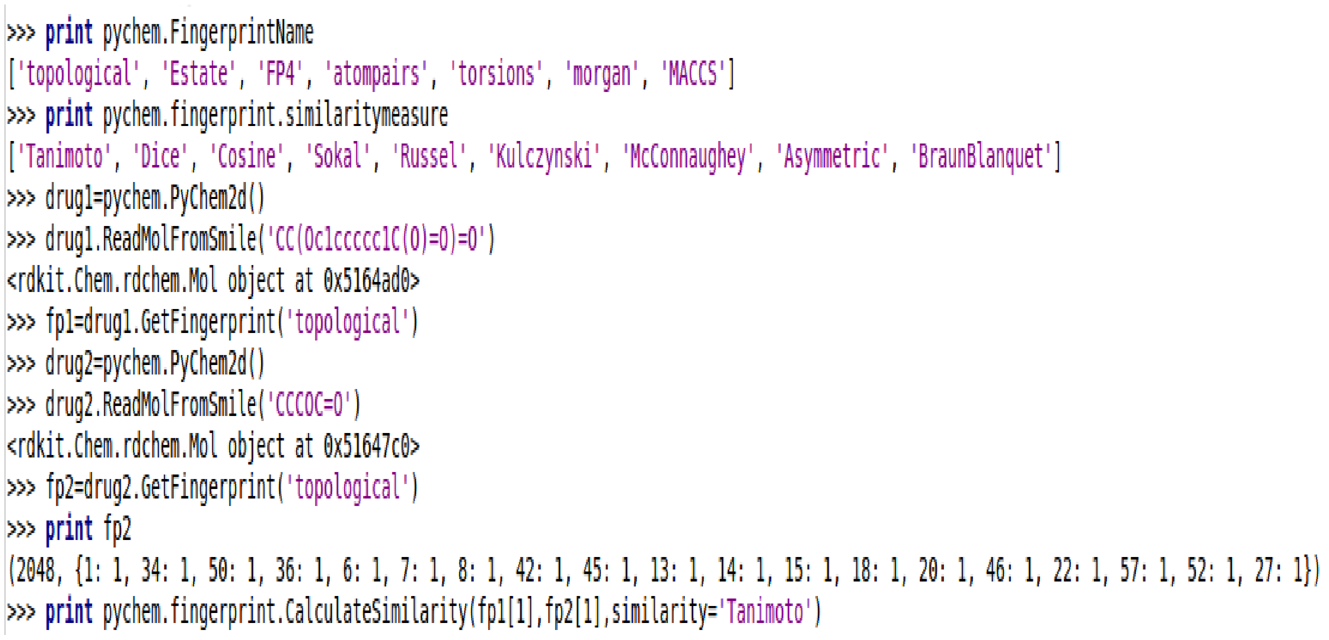
The convenient way to calculate the fingerprints is to generate a PyDrug object and call GetFingerprint method.



```
>>> smi='CC(Oc1ccccc1C(O)=O)=O'
>>> drug=pychem.PyChem2d()
>>> drug.ReadMolFromSmile(smi)
<rdkit.Chem.rdchem.Mol object at 0x5164750>
>>> print pychem.FingerprintName
['topological', 'Estate', 'FP4', 'atompairs', 'torsions', 'morgan', 'MACCS']
>>> fp1=drug.GetFingerprint('topological')
>>> fp2=drug.GetFingerprint('Estate')
>>> fp3=drug.GetFingerprint('MACCS')
>>> print fp1[0], fp2[0], fp3[0]
2048 79 166
```

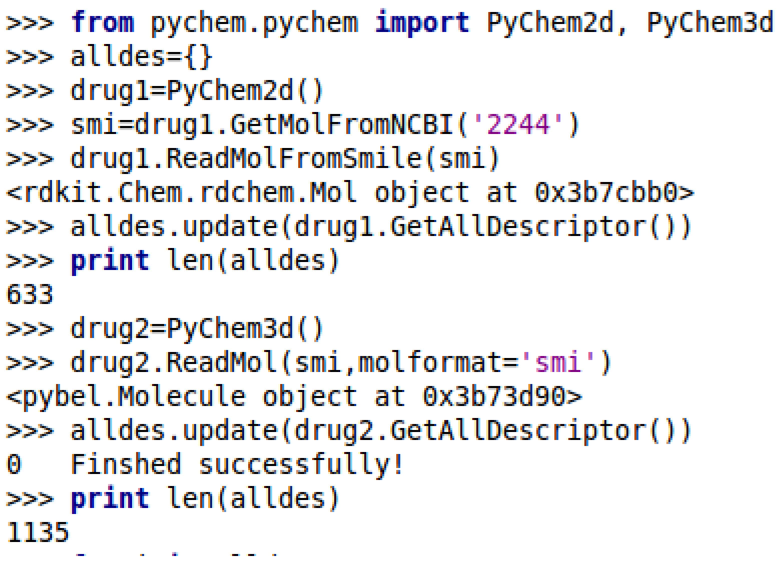
5.3.7. fingerprint similarity

We could any fingerprint similarity using the nine given similarity measure methods.



```
>>> print pychem.FingerprintName
['topological', 'Estate', 'FP4', 'atompairs', 'torsions', 'morgan', 'MACCS']
>>> print pychem.fingerprint.similaritymeasure
['Tanimoto', 'Dice', 'Cosine', 'Sokal', 'Russel', 'Kulczynski', 'McConnaughey', 'Asymmetric', 'BraunBlanquet']
>>> drug1=pychem.PyChem2d()
>>> drug1.ReadMolFromSmile('CC(Oc1ccccc1C(O)=O)=O')
<rdkit.Chem.rdchem.Mol object at 0x5164ad0>
>>> fp1=drug1.GetFingerprint('topological')
>>> drug2=pychem.PyChem2d()
>>> drug2.ReadMolFromSmile('CCCOC=O')
<rdkit.Chem.rdchem.Mol object at 0x51647c0>
>>> fp2=drug2.GetFingerprint('topological')
>>> print fp2
(2048, {1: 1, 34: 1, 50: 1, 36: 1, 6: 1, 7: 1, 8: 1, 42: 1, 45: 1, 13: 1, 14: 1, 15: 1, 18: 1, 20: 1, 46: 1, 22: 1, 57: 1, 52: 1, 27: 1})
>>> print pychem.fingerprint.CalculateSimilarity(fp1[1],fp2[1],similarity='Tanimoto')
```

6. Calculating all descriptors



```
>>> from pychem.pychem import PyChem2d, PyChem3d
>>> alldes={}
>>> drug1=PyChem2d()
>>> smi=drug1.GetMolFromNCBI('2244')
>>> drug1.ReadMolFromSmile(smi)
<rdkit.Chem.rdchem.Mol object at 0x3b7cbb0>
>>> alldes.update(drug1.GetAllDescriptor())
>>> print len(alldes)
633
>>> drug2=PyChem3d()
>>> drug2.ReadMol(smi,molformat='smi')
<pybel.Molecule object at 0x3b73d90>
>>> alldes.update(drug2.GetAllDescriptor())
0   Finshed successfully!
>>> print len(alldes)
1135
```

Appendix:

Table S1: List of PyChem computed molecular descriptors

Molecular descriptors		
	Constitutional descriptors (30)	
1	Weight	Molecular weight
2	nhyd	Count of hydrogen atoms
3	nhal	Count of halogen atoms
4	nhet	Count of hetero atoms
5	nhev	Count of heavy atoms
6	ncof	Count of F atoms
7	ncocl	Count of Cl atoms
8	ncobr	Count of Br atoms
9	ncoi	Count of I atoms
10	ncarb	Count of C atoms
11	nphos	Count of P atoms
12	nsulph	Count of S atoms
13	noxy	Count of O atoms
14	nnitro	Count of N atoms
15	nring	Number of rings

16	nrot	Number of rotatable bonds
17	ndonr	Number of H-bond donors
18	naccr	Number of H-bond acceptors
19	nsb	Number of single bonds
20	ndb	Number of double bonds
21	ntb	Number of triple bonds
22	naro	Number of aromatic bonds
23	nta	Number of all atoms
24	AWeight	Average molecular weight
25-30	PC1 PC2 PC3 PC4 PC5 PC6	Molecular path counts of length 1-6
	Topological descriptors (35)	
1	W	Weiner index
2	AW	Average Wiener index
3	J	Balaban’s J index
4	T_{hara}	Harary number
5	T_{sch}	Schiultz index
6	Tigdi	Graph distance index
7	Platt	Platt number

8	Xu	Xu index
9	Pol	Polarity number
10	Dz	Pogliani index
11	Ipc	Ipc index
12	BertzCT	BertzCT
13	GMTI	Gutman molecular topological index based on simple vertex degree
14-15	ZM1 ZM2	Zagreb index with order 1-2
16-17	MZM1 MZM2	Modified Zagreb index with order 1-2
18	Qindex	Quadratic index
19	diametert	Largest value in the distance matrix
20	radiust	radius based on topology
21	petitjeant	Petitjean based on topology
22	Sito	the logarithm of the simple topological index by Narumi
23	Hato	harmonic topological index proposed by Narnumi
24	Geto	Geometric topological index by Narumi
25	Arto	Arithmetic topological index by Narumi
26	ISIZ	Total information index on molecular size
27	TIAC	Total information index on atomic composition
28	DET	Total information index on distance equality
29	IDE	Mean information index on distance equality

30	IVDE	Total information index on vertex equality
31	Sitov	Logarithm of the simple topological index by Narumi
32	Hatov	Harmonic topological index proposed by Narnumi
33	Getov	Geometric topological index by Narumi
34	Gravto	Gravitational topological index based on topological distance
35	GMTIV	Gutman molecular topological index based on valence vertex degree(log10)
	Connectivity descriptors (44)	
1-11	${}^{0}\chi^{v}$ ${}^{1}\chi^{v}$ ${}^{2}\chi^{v}$ ${}^{3}\chi_{p}^{v}$ ${}^{4}\chi_{p}^{v}$ ${}^{5}\chi_{p}^{v}$ ${}^{6}\chi_{p}^{v}$ ${}^{7}\chi_{p}^{v}$ ${}^{8}\chi_{p}^{v}$ ${}^{9}\chi_{p}^{v}$ ${}^{10}\chi_{p}^{v}$	Valence molecular connectivity Chi index for path order 0-10
12	${}^{3}\chi^{v}_{c}$	Valence molecular connectivity Chi index for three cluster
13	${}^{4}\chi^{v}_{c}$	Valence molecular connectivity Chi index for four cluster
14	${}^{4}\chi^{v}_{pc}$	Valence molecular connectivity Chi index for path/cluster
15-18	${}^{3}\chi^{v}_{CH}$ ${}^{4}\chi^{v}_{CH}$ ${}^{5}\chi^{v}_{CH}$ ${}^{6}\chi^{v}_{CH}$	Valence molecular connectivity Chi index for cycles of 3-6
19-29	${}^{0}\chi$ ${}^{1}\chi$ ${}^{2}\chi$ ${}^{3}\chi_{p}$ ${}^{4}\chi_{p}$ ${}^{5}\chi_{p}$ ${}^{6}\chi_{p}$ ${}^{7}\chi_{p}$	Simple molecular connectivity Chi indices for path order 0-10

	$^{8}\chi_{p}$ $^{9}\chi_{p}$ $^{10}\chi_{p}$	
30	$^{3}\chi_{c}$	Simple molecular connectivity Chi indices for three cluster
31	$^{4}\chi_{c}$	Simple molecular connectivity Chi indices for four cluster
32	$^{4}\chi_{pc}$	Simple molecular connectivity Chi indices for path/cluster
33-36	$^{3}\chi_{CH}$ $^{4}\chi_{CH}$ $^{5}\chi_{CH}$ $^{6}\chi_{CH}$	Simple molecular connectivity Chi indices for cycles of 3-6
37	mChi1	mean chi1 (Randic) connectivity index
38	knotp	the difference between chi3c and chi4pc
39	dchi0	the difference between chi0v and chi0
40	dchi1	the difference between chi1v and chi1
41	dchi2	the difference between chi2v and chi2
42	dchi3	the difference between chi3v and chi3
43	dchi4	the difference between chi4v and chi4
44	knotpv	the difference between chiv3c and chiv4pc
	Kappa descriptors (7)	
1	$^{1}\kappa_{\alpha}$	Kappa alpha index for 1 bonded fragment
2	$^{2}\kappa_{\alpha}$	Kappa alpha index for 2 bonded fragment
3	$^{3}\kappa_{\alpha}$	Kappa alpha index for 3 bonded fragment
4	phi	Kier molecular flexibility index
5	$^{1}\kappa$	Molecular shape Kappa index for 1 bonded fragment

6	$^{2}\kappa$	Molecular shape Kappa index for 2 bonded fragment
7	$^{3}\kappa$	Molecular shape Kappa index for 3 bonded fragment
	Basak descriptors (21)	
1	IC0	Information content with order 0 proposed by Basak
2	IC1	Information content with order 1 proposed by Basak
3	IC2	Information content with order 2 proposed by Basak
4	IC3	Information content with order 3 proposed by Basak
5	IC4	Information content with order 4 proposed by Basak
6	IC5	Information content with order 5 proposed by Basak
7	IC6	Information content with order 6 proposed by Basak
8	SIC0	Complementary information content with order 0 proposed by Basak
9	SIC1	Structural information content with order 1 proposed by Basak
10	SIC2	Structural information content with order 2 proposed by Basak
11	SIC3	Structural information content with order 3 proposed by Basak
12	SIC4	Structural information content with order 4 proposed by Basak
13	SIC5	Structural information content with order 5 proposed by Basak
14	SIC6	Structural information content with order 6 proposed by Basak
15	CIC0	Complementary information content with order 0 proposed by Basak
16	CIC1	Complementary information content with order 1 proposed by Basak
17	CIC2	Complementary information content with order 2 proposed by Basak

18	CIC3	Complementary information content with order 3 proposed by Basak
19	CIC4	Complementary information content with order 4 proposed by Basak
20	CIC5	Complementary information content with order 5 proposed by Basak
21	CIC6	Complementary information content with order 6 proposed by Basak
	E-state descriptors (245)	
1	S1	Sum of E-State of atom type: sLi
2	S2	Sum of E-State of atom type: ssBe
3	S3	Sum of E-State of atom type: ssssBe
4	S4	Sum of E-State of atom type: ssBH
5	S5	Sum of E-State of atom type: sssB
6	S6	Sum of E-State of atom type: ssssB
7	S7	Sum of E-State of atom type: sCH3
8	S8	Sum of E-State of atom type: dCH2
9	S9	Sum of E-State of atom type: ssCH2
10	S10	Sum of E-State of atom type: tCH
11	S11	Sum of E-State of atom type: dsCH
12	S12	Sum of E-State of atom type: aaCH
13	S13	Sum of E-State of atom type: sssCH
14	S14	Sum of E-State of atom type: ddC
15	S15	Sum of E-State of atom type: tsC

16	S16	Sum of E-State of atom type: dssC
17	S17	Sum of E-State of atom type: aasC
18	S18	Sum of E-State of atom type: aaaC
19	S19	Sum of E-State of atom type: ssssC
20	S20	Sum of E-State of atom type: sNH3
21	S(21)	Sum of E-State of atom type: sNH2
22	S22	Sum of E-State of atom type: ssNH2
23	S23	Sum of E-State of atom type: dNH
24	S24	Sum of E-State of atom type: ssNH
25	S25	Sum of E-State of atom type: aaNH
26	S26	Sum of E-State of atom type: tN
27	S27	Sum of E-State of atom type: sssNH
28	S28	Sum of E-State of atom type: dsN
29	S29	Sum of E-State of atom type: aaN
30	S30	Sum of E-State of atom type: sssN
31	S31	Sum of E-State of atom type: ddsN
32	S32	Sum of E-State of atom type: aasN
33	S33	Sum of E-State of atom type: ssssN
34	S34	Sum of E-State of atom type: sOH
35	S35	Sum of E-State of atom type: dO

36	S36	Sum of E-State of atom type: ssO
37	S37	Sum of E-State of atom type: aaO
38	S38	Sum of E-State of atom type: sF
39	S39	Sum of E-State of atom type: sSiH3
40	S40	Sum of E-State of atom type: ssSiH2
41	S41	Sum of E-State of atom type: sssSiH
42	S42	Sum of E-State of atom type: ssssSi
43	S43	Sum of E-State of atom type: sPH2
44	S44	Sum of E-State of atom type: ssPH
45	S45	Sum of E-State of atom type: sssP
46	S46	Sum of E-State of atom type: dsssP
47	S47	Sum of E-State of atom type: sssssP
48	S48	Sum of E-State of atom type: sSH
49	S49	Sum of E-State of atom type: dS
50	S50	Sum of E-State of atom type: ssS
51	S51	Sum of E-State of atom type: aaS
52	S52	Sum of E-State of atom type: dssS
53	S53	Sum of E-State of atom type: ddssS
54	S54	Sum of E-State of atom type: sCl
55	S55	Sum of E-State of atom type: sGeH3

56	S56	Sum of E-State of atom type: ssGeH2
57	S57	Sum of E-State of atom type: sssGeH
58	S58	Sum of E-State of atom type: ssssGe
59	S59	Sum of E-State of atom type: sAsH2
60	S60	Sum of E-State of atom type: ssAsH
61	S61	Sum of E-State of atom type: sssAs
62	S62	Sum of E-State of atom type: sssdAs
63	S63	Sum of E-State of atom type: sssssAs
64	S64	Sum of E-State of atom type: sSeH
65	S65	Sum of E-State of atom type: dSe
66	S66	Sum of E-State of atom type: ssSe
67	S67	Sum of E-State of atom type: aaSe
68	S68	Sum of E-State of atom type: dssSe
69	S69	Sum of E-State of atom type: ddssSe
70	S70	Sum of E-State of atom type: sBr
71	S71	Sum of E-State of atom type: sSnH3
72	S72	Sum of E-State of atom type: ssSnH2
73	S73	Sum of E-State of atom type: sssSnH
74	S74	Sum of E-State of atom type: ssssSn
75	S75	Sum of E-State of atom type: sI

76	S76	Sum of E-State of atom type: sPbH3
77	S77	Sum of E-State of atom type: ssPbH2
78	S78	Sum of E-State of atom type: sssPbH
79	S79	Sum of E-State of atom type: ssssPb
80-158	Smax1-Smax79	Maxmum of E-State value of specified atom type
159-237	Smin1-Smin79	Minimum of E-State value of specified atom type
238	Shev	The sum of the EState indices over all non-hydrogen atoms
239	Scar	The sum of the EState indices over all C atoms
240	Shal	The sum of the EState indices over all Halogen atoms
241	Shet	The sum of the EState indices over all hetero atoms
242	Save	The sum of the EState indices over all non-hydrogen atoms divided by the number of non-hydrogen atoms
243	Smax	The maximal Estate value in all atoms
244	Smin	The minimal Estate value in all atoms
245	DS	The difference between Smax and Smin
	Burden descriptors (64)	
1-16	bcutm1-16	Burden descriptors based on atomic mass
17-32	bcutv1-16	Burden descriptors based on atomic vloumes
33-48	bcute1-16	Burden descriptors based on atomic electronegativity
49-64	bcutp1-16	Burden descriptors based on polarizability
	Autocorrelation descriptors (96)	

1-8	ATSm1-ATSm8	Moreau-Broto autocorrelation descriptors based on atom mass
9-16	ATSv1-ATSv8	Moreau-Broto autocorrelation descriptors based on atomic van der Waals volume
17-24	ATSe1-ATSe8	Moreau-Broto autocorrelation descriptors based on atomic Sanderson electronegativity
25-32	ATSp1-ATSp8	Moreau-Broto autocorrelation descriptors based on atomic polarizability
33-40	MATSm1-MATSm8	Moran autocorrelation descriptors based on atom mass
41-48	MATSv1-MATSv8	Moran autocorrelation descriptors based on atomic van der Waals volume
49-56	MATSe1-MATSe8	Moran autocorrelation descriptors based on atomic Sanderson electronegativity
57-64	MATSp1-MATSp8	Moran autocorrelation descriptors based on atomic polarizability
65-72	GATSm1-GATSm8	Geary autocorrelation descriptors based on atom mass
73-80	GATSv1-GATSv8	Geary autocorrelation descriptors based on atomic van der Waals volume
81-88	GATSe1-GATSe8	Geary autocorrelation descriptors based on atomic Sanderson electronegativity
89-96	GATSp1-GATSp8	Geary autocorrelation descriptors based on atomic polarizability
	Charge descriptors (25)	
1-4	Q_{Hmax} Q_{Cmax} Q_{Nmax} Q_{Omax}	Most positive charge on H,C,N,O atoms
5-8	Q_{Hmin} Q_{Cmin} Q_{Nmin} Q_{Omin}	Most negative charge on H,C,N,O atoms
9-10	Q_{max} Q_{min}	Most positive and negative charge in a molecule
11-15	Q_{HSS} Q_{CSS} Q_{NSS} Q_{OSS} Qass	Sum of squares of charges on H,C,N,O and all toms

16-17	Mpc Tpc	Mean and total of positive charges
18-19	Mnc Tnc	Mean and total of negative charges
20-21	Mac Tac	Mean and total of absolute charges
22	Rpc	Relative positive charge
23	Rnc	Relative negative charge
24	SPP	Submolecular polarity parameter
25	LDI	Local dipole index
	Molecular property descriptors (6)	
1	MREF	Molar refractivity
2	logP	LogP value based on the Crippen method
3	logP2	Square of LogP value based on the Crippen method
4	TPSA	Topological polarity surface area
5	UI	Unsaturation index
6	Hy	Hydrophilic index
	MOE-type descriptors (60)	
1	TPSA	Topological polar surface area based on fragments
2	LabuteASA	Labute's Approximate Surface Area
3-14	SLOGPVSA	MOE-type descriptors using SLogP contributions and surface area contributions
15-24	SMRVSA	MOE-type descriptors using MR contributions and surface area contributions
25-38	PEOEVSA	MOE-type descriptors using partial charges and surface area contributions

39-49	EstateVSA	MOE-type descriptors using Estate indices and surface area contributions
50-60	VSAEstate	MOE-type descriptors using surface area contributions and Estate indices
	Geometric descriptors (12)	
1	W3DH	3-D Wiener index based geometrical distance matrix (including Hs)
2	W3D	3-D Wiener index based geometrical distance matrix (Not including Hs)
3	Petitj3D	Petitjean Index based on molecular geometrical distance matrix
4	GeDi	The longest distance between two atoms (geometrical diameter)
5	grav1	Gravitational 3D index
6	rygr	Radius of gyration
7	Harary3D	The 3D-Harary index
8	AGDD	The average geometric distance degree
9	SEig	The absolute eigenvalue sum on geometry matrix
10	SPAN	The span R
11	ASPAN	The average span R
12	MEcc	The molecular eccentricity
	CPSA descriptors (30)	
1	ASA	Solvent-accessible surface areas
2	MSA	Molecular surface areas
3	PNSA1	Partial negative area
4	PPSA1	Partial negative area

5	PNSA2	Total charge weighted negative surface area
6	PPSA2	Total charge weighted negative surface area
7	PNSA3	Atom charge weighted negative surface areas
8	PPSA3	Atom charge weighted positive surface areas
9	DPSA1	Difference in charged partial surface area
10	DPSA2	Difference in total charge weighted partial surface area
11	DPSA3	Difference in atomic charge weighted surface area
12	FNSA1	Fractional charged partial negative surface areas
13	FNSA2	Fractional charged partial negative surface areas
14	FNSA3	Fractional charged partial negative surface areas
15	FPSA1	Fractional charged partial negative surface areas
16	FPSA2	Fractional charged partial negative surface areas
17	FPSA3	Fractional charged partial negative surface areas
18	WNSA1	Surface weighted charged partial negative surface areas
19	WNSA2	Surface weighted charged partial negative surface areas
20	WNSA3	Surface weighted charged partial negative surface areas
21	WPSA1	Surface weighted charged partial negative surface areas
22	WPSA2	Surface weighted charged partial negative surface areas
23	WPSA3	Surface weighted charged partial negative surface areas
24	TASA	Total hydrophobic surface area

25	PSA	Total polar surface area
26	FrTATP	The fraction between TASA and TPSA
27	RASA	Relative hydrophobic surface area
28	RPSA	Relative polar surface area
29	RNCS	Relative negative charge surface area
30	RPCS	Relative positive charge surface area
	WHIM descriptors (70)	
1-14	---	Unweighted WHIM descriptors
15-28	---	WHIM descriptors based on atomic mass
29-42	---	WHIM descriptors based on Sanderson Electronegativity
43-56	---	WHIM descriptors based on VDW Volume
57-70	---	WHIM descriptors based on Polarizability
	MoRSE descriptors (210)	
1-30	MoRSEU1-30	Unweighted 3-D MoRse descriptors
31-60	MoRSEC1-30	3-D MoRse descriptors based on atomic charge
61-90	MoRSEM1-30	3-D MoRse descriptors based on atomic mass
91-120	MoRSEN1-30	3-D MoRse descriptors based on atomic number
121-150	MoRSEP1-30	3-D MoRse descriptors based on atomic polarizablity
151-180	MoRSEE1-30	3-D MoRse descriptors based on atomic Sanderson electronegativity
181-210	MoRSEV1-30	3-D MoRse descriptors based on atomic van der Waals volume

	RDF descriptors (180)	
1-30	RDFU1-30	Unweighted radial distribution function (RDF) descriptors
31-60	RDFC1-30	Radial distribution function (RDF) descriptors based on atomic charge.
61-90	RDFM1-30	Radial distribution function (RDF) descriptors based on atomic mass
91-120	RDFP1-30	Radial distribution function (RDF) descriptors based on atomic polarizability
121-150	RDFE1-30	Radial distribution function (RDF) descriptors based on atomic electronegativity
151-180	RDFV1-30	Radial distribution function (RDF) descriptors based on atomic van der Waals volume
	Fragment/Fingerprint-based descriptors	
1	FP2	(Topological fingerprint) A Daylight-like fingerprint based on hashing molecular subgraphs
2	MACCS	(MACCS keys)Using the 166 public keys implemented as SMARTS
3	E-state	79 E-state fingerprints or fragments
4	FP4	307 FP4 fingerprints
5	Atom Paris	Atom Paris fingerprints
6	Torsions	Topological torsion fingerprints
7	Morgan/Circular	Fingerprints based on the Morgan algorithm