
NAME

Molecule - Molecule class

SYNOPSIS

use Molecule;

use Molecule qw(:all);

DESCRIPTION

Molecule class provides the following methods:

new, AddAtom, AddAtoms, AddBond, AddBonds, AddHydrogens, AddPolarHydrogens, ClearRings, Copy, DeleteAromaticity, DeleteAtom, DeleteAtoms, DeleteBond, DeleteBonds, DeleteHydrogens, DeletePolarHydrogens, DetectAromaticity, DetectRings, FormatElementalCompositionInformation, GetAllAtomPaths, GetAllAtomPathsStartingAt, GetAllAtomPathsStartingAtWithLengthUpto, GetAllAtomPathsWithLengthUpto, GetAtomNeighborhoods, GetAtomNeighborhoodsWithRadiusUpto, GetAtomPathBonds, GetAtomPaths, GetAtomPathsBetween, GetAtomPathsStartingAt, GetAtomPathsStartingAtWithLengthUpto, GetAtomPathsWithLengthUpto, GetAtoms, GetBonds, GetCharge, GetConnectedComponents, GetConnectedComponentsAtoms, GetElementalComposition, GetElementsAndNonElements, GetExactMass, GetFormalCharge, GetFusedAndNonFusedRings, GetLargestConnectedComponent, GetLargestConnectedComponentAtoms, GetLargestRing, GetMolecularFormula, GetMolecularWeight, GetNumOfAtoms, GetNumOfBonds, GetNumOfConnectedComponents, GetNumOfHeavyAtoms, GetNumOfHydrogenAtoms, GetNumOfMissingHydrogenAtoms, GetNumOfNonHydrogenAtoms, GetNumOfRings, GetNumOfRingsWithEvenSize, GetNumOfRingsWithOddSize, GetNumOfRingsWithSize, GetNumOfRingsWithSizeGreaterThan, GetNumOfRingsWithSizeLessThan, GetRingBonds, GetRings, GetRingsWithEvenSize, GetRingsWithOddSize, GetRingsWithSize, GetRingsWithSizeGreaterThan, GetRingsWithSizeLessThan, GetSizeOfLargestRing, GetSizeOfSmallestRing, GetSmallestRing, GetSpinMultiplicity, GetTopologicallySortedAtoms, HasAtom, HasBond, HasFusedRings, HasNoRings, HasOnlyOneRing, HasRings, IsMolecule, IsThreeDimensional, KeepLargestComponent, NewAtom, NewBond, SetActiveRings, StringifyMolecule

The following methods can also be used as functions:

FormatElementalCompositionInformation, IsMolecule

Molecule class is derived from ObjectProperty base class which provides methods not explicitly defined in Molecule or ObjectProperty class using Perl's AUTOLOAD functionality. These methods are generated on-the-fly for a specified object property:

```
Set<PropertyName>(<PropertyValue>);
$PropertyValue = Get<PropertyName>();
Delete<PropertyName>();
```

METHODS**new**

```
$NewMolecule = new Molecule([%PropertyNameAndValues]);
```

Using specified *Atom* property names and values hash, new method creates a new object and returns a reference to newly created Atom object. By default, the following properties are initialized:

```
ID = SequentialObjectID
Name = "Molecule <SequentialObjectID>"
```

Examples:

```
$Molecule = new Molecule();

$WaterMolecule = new Molecule('Name' => 'Water');

$Oxygen = new Atom('AtomSymbol' => 'O', 'XYZ' => [0, 0, 0]);
$Hydrogen1 = new Atom('AtomSymbol' => 'H',
                    'XYZ' => [0.7144, 0.4125, 0]);
$Hydrogen2 = new Atom('AtomSymbol' => 'H',
                    'XYZ' => [1.1208, -0.2959, 0]);
$WaterMolecule->AddAtoms($Oxygen, $Hydrogen1, $Hydrogen2);
```

```
$Bond1 = new Bond('Atoms' => [$Oxygen, $Hydrogen1],  
                 'BondOrder' => 1);  
$Bond2 = new Bond('Atoms' => [$Oxygen, $Hydrogen2],  
                 'BondOrder' => 1);  
$WaterMolecule->AddBonds($Bond1, $Bond2);
```

AddAtom

```
$Molecule->AddAtom($Atom);
```

Adds an *Atom* to a *Molecule* and returns *Molecule*

AddAtoms

```
$Molecule->AddAtoms(@Atoms);
```

Adds *Atoms* to a *Molecule* and returns *Molecule*

AddBond

```
$Molecule->AddBond($Bond);
```

Adds a *Bond* to a *Molecule* and returns *Molecule*

AddBonds

```
$Molecule->AddBonds(@Bonds);
```

Adds *Bonds* to a *Molecule* and returns *Molecule*

AddHydrogens

```
$NumOfHydrogensAdded = $Molecule->AddHydrogens();
```

Adds hydrogens to each atom in a *Molecule* and returns total number of hydrogens added. The current release of MayaChemTools doesn't assign hydrogen positions.

AddPolarHydrogens

```
$NumOfHydrogensAdded = $Molecule->AddPolarHydrogens();
```

Adds hydrogens to each polar atom - N, O, P or S - in a *Molecule* and returns total number of polar hydrogens added. The current release of MayaChemTools doesn't assign hydrogen positions.

ClearRings

```
$Molecule->ClearRings();
```

Deletes all rings associated with *Molecule* and returns *Molecule*

Copy

```
$MoleculeCopy = $Molecule->Copy();
```

Copies *Molecule* and its associated data using `Storable::dclone` and returns a new *Molecule* object

DeleteAromaticity

```
$Molecule->DeleteAromaticity();
```

Deletes aromatic property associated with all atoms and bonds in a *Molecule* and returns *Molecule*

DeleteAtom

```
$Molecule->DeleteAtom($Atom);
```

Deletes *Atom* from a *Molecule* and returns *Molecule*

DeleteAtoms

```
$Molecule->DeleteAtoms(@Atoms);
```

Deletes *Atoms* from a *Molecule* and returns *Molecule*

DeleteBond

```
$Molecule->DeleteBond($Bond);
```

Deletes *Bond* from a *Molecule* and returns *Molecule*

DeleteBonds

```
$Molecule->DeleteBonds(@Bonds);
```

Deletes *Bonds* from a *Molecule* and returns *Molecule*

DeleteHydrogens

```
$NumOfHydrogensDeleted = $Molecule->DeleteHydrogens();
```

Removes hydrogens from each atom in a *Molecule* and returns total number of hydrogens deleted

DeletePolarHydrogens

```
$NumOfHydrogensDeleted = $Molecule->DeletePolarHydrogens();
```

Removes hydrogens to each polar atom - N, O, P or S - in a *Molecule* and returns total number of polar hydrogens deleted

DetectAromaticity

```
$Molecule->DetectAromaticity();
```

Associates *Aromatic* property to atoms and bonds involved in aromatic rings in a *Molecule* and returns *Molecule*.

This method assumes the ring detection has already been performed using DetectRings. And any existing *Aromatic* property associated with atoms and bonds is deleted before performing aromaticity detection.

Notes:

o Ring aromaticity is determined using Huckel's rule: a ring containing $4n + 2$ pi electrons is considered aromatic.

o Heterocyclic rings containing N, O and S atoms fall into two classes: Basic aromatic and Non-basic aromatic. In Basic aromatic heterocyclic rings, heteroatom itself is involved in a double bond. (e.g. Pyridine) However, in non-basic heterocyclic rings, heteroatom might have an attached hydrogen atom and the remaining lone pair contribute to electron delocalization and contributes to $4n + 2$ electrons. (e.g. Pyrrole, Furan)

o For molecules containing fused rings, each fused ring set is considered as one aromatic system for counting pi electrons to satisfy Huckel's rule; In case of a failure, rings in fused set are treated individually for aromaticity detection. Additionally, non-fused rings are handled on their own during aromaticity detection.

DetectRings

```
$Molecule->DetectRings();
```

Detects rings in a *Molecule* and returns *Molecule*. Ring detection is performed using DetectCycles method available in Graph class which in turn uses methods available CyclesDetection class. CyclesDetection class implements collapsing path graph [Ref 31] methodology to detect all cycles in a graph.

FormatElementalCompositionInformation

```
$FormattedInfo = $Molecule->FormatElementalCompositionInformation(
    $ElementsRef, $ElementCompositionRef,
    [$Precision]);
$FormattedInfo = Molecule::FormatElementalCompositionInformation(
    $ElementsRef, $ElementCompositionRef,
    [$Precision]);
```

Using *ElementsRef* and *ElementCompositionRef* arrays references containing information about elements and their composition, formats elemental composition information and returns a *FormattedInfo* string. Default *Precision*

value: 2.

GetAllAtomPaths

```
$AtomPathsRef = $Molecule->GetAllAtomPaths([$AllowCycles]);
```

Returns all paths as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to to all possible paths for each atom in molecule with all possible lengths and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as GetAtomPaths method.

GetAllAtomPathsStartingAt

```
$AtomPathsRef = $Molecule->GetAllAtomPathsStartingAt($StartAtom,
[$AllowCycles]);
```

Returns all atom paths starting from *StartAtom* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms atoms correspond to to all possible paths for specified atom in molecule with all possible lengths and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as GetAtomPathsStartingAt method.

GetAllAtomPathsStartingAtWithLengthUpto

```
$AtomPathsRef = $Molecule->GetAllAtomPathsStartingAtWithLengthUpto(
$StartAtom, $Length, [$AllowCycles]);
```

Returns atom paths starting from *StartAtom* with length up to *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms atoms correspond to all possible paths for specified atom in molecule with length up to a specified length and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as *GetAtomPathsStartingAtWithLengthUpto* method.

GetAllAtomPathsWithLengthUpto

```
$AtomPathsRef = $Molecule->GetAllAtomPathsWithLengthUpto($Length,
[$AllowCycles]);
```

Returns all atom paths with length up to *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to to all possible paths for each atom in molecule with length up to a specified length and sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

For molecule without any rings, this method returns the same set of atom paths as as *GetAtomPathsWithLengthUpto* method.

GetAtomNeighborhoods

```
$AtomsRef = $Molecule->GetAtomNeighborhoods($StartAtom);
```

Returns atom neighborhoods around a *StartAtom* as an array containing references to arrays with neighborhood *Atom* objects at possible radii

GetAtomNeighborhoodsWithRadiusUpto

```
$AtomsRef = $Molecule->GetAtomNeighborhoodsWithRadiusUpto($StartAtom,
$Radius);
```

Returns atom neighborhoods around a *StartAtom* as an array containing references to arrays with

neighborhood *Atom* objects up to *Radius*

GetAtomPathBonds

```
$Return = $Molecule->GetAtomPathBonds(@PathAtoms);
```

Returns an array containing Bond objects corresponding to successive pair of atoms in *PathAtoms*

GetAtomPaths

```
$AtomPathsRef = $Molecule->GetAtomPaths([$AllowCycles]);
```

Returns all paths as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to to all possible paths for each atom in molecule with all possible lengths and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtomPathsBetween

```
$AtomPathsRef = $Molecule->GetAtomPathsBetween($StartAtom, $EndAtom);
```

Returns all paths as between *StartAtom* and *EndAtom* as a reference to an array containing reference to arrays with path Atom objects.

For molecules with rings, atom paths array contains may contain two paths.

GetAtomPathsStartingAt

```
$AtomPathsRef = $Molecule->GetAtomPathsStartingAt($StartAtom, [$AllowCycles]);
```

Returns paths starting at *StartAtom* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to all possible paths for specified atom in molecule with all possible lengths and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtomPathsStartingAtWithLengthUpto

```
$AtomPathsRef = $Molecule->GetAtomPathsStartingAtWithLengthUpto($StartAtom,  
    $Length, $AllowCycles);
```

Returns paths starting at *StartAtom* with length up to *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to all possible paths for specified atom in molecule with length upto a specified length and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtomPathsWithLengthUpto

```
$AtomPathsRef = $Molecule->GetAtomPathsWithLengthUpto($Length, [$AllowCycles]);
```

Returns all paths with length up to *Length* as a reference to an array containing reference to arrays with path Atom objects.

Path atoms correspond to all possible paths for each atom in molecule with length upto a specified length and no sharing of bonds in paths traversed. By default, rings are included in paths. A path containing a ring is terminated at an atom completing the ring.

GetAtoms

```
@Atoms = $Molecule->GetAtoms();  
@Atoms = $Molecule->GetAtoms('IsPolarAtom');  
$AtomsCount = $Molecule->GetAtoms();
```

Returns an array of *Atoms* in a *Molecule*. In scalar context, it returns number of atoms. Additionally, Atoms array can be filtered by any user specifiable valid Atom class method.

GetBonds

```
@Bonds = $Molecule->GetBonds();
```

```
$BondsCount = $Molecule->GetBonds();
```

Returns an array of *Bonds* in a *Molecule*. In scalar context, it returns number of bonds

GetCharge

```
$Charge = $Molecule->GetCharge();
```

Returns net charge on a *Molecule* using one of the following two methods: explicitly set Charge property or sum of partial atomic charges on each atom.

GetConnectedComponents

```
@ConnectedComponents = $Molecule->GetConnectedComponents();
```

Returns a reference to an array containing *Molecule* objects corresponding to connected components sorted in decreasing order of component size in a *Molecule*

GetConnectedComponentsAtoms

```
@ConnectedComponentsAtoms =  
$Molecule->GetConnectedComponentsAtoms();
```

Returns an array containing references to arrays with *Atom* objects corresponding to atoms of connected components sorted in order of component decreasing size in a *Molecule*

GetElementalComposition

```
($ElementsRef, $CompositionRef) =  
$Molecule->GetElementalComposition([$IncludeMissingHydrogens]);
```

Calculates elemental composition and returns references to arrays containing elements and their percent composition in a *Molecule*. By default, missing hydrogens are included during the calculation.

GetElementsAndNonElements

```
($ElementsRef, $NonElementsRef) =  
$Molecule->GetElementsAndNonElements([$IncludeMissingHydrogens]);
```

Counts elements and non-elements in a *Molecule* and returns references to hashes containing element and non-element as hash keys with values corresponding to their count. By default, missing hydrogens are not added to the element hash.

GetExactMass

```
$ExactMass = $Molecule->GetExactMass();
```

Returns exact mass of a *Molecule* corresponding to sum of exact masses of all the atoms

GetFormalCharge

```
$FormalCharge = $Molecule->GetFormalCharge();
```

Returns net formal charge on a *Molecule* using one of the following two methods: explicitly set FormalCharge property or sum of formal charges on each atom.

FormalCharge is different from Charge property of the molecule which corresponds to sum of partial atomic charges explicitly set for each atom using a specific methodology.

GetFusedAndNonFusedRings

```
($FusedRingSetRef, $NonFusedRingsRef) =  
$Molecule->GetFusedAndNonFusedRings();
```

Returns references to array of fused ring sets and non-fused rings in a *Molecule*. Fused ring sets array reference contains references to arrays of rings corresponding to ring *Atom* objects; Non-fused rings array reference contains references to arrays of ring *Atom* objects.

GetLargestConnectedComponent

```
$ComponentMolecule = $Molecule->GetLargestConnectedComponent();
```

Returns a reference to Molecule object corresponding to a largest connected component in a *Molecule*

GetLargestConnectedComponentAtoms

```
@ComponentAtoms = $Molecule->GetLargestConnectedComponentAtoms();
```

Returns a reference to an array of Atom objects corresponding to a largest connected component in a *Molecule*

GetLargestRing

```
@RingAtoms = $Molecule->GetLargestRing();
```

Returns an array of *Atoms* objects corresponding to a largest ring in a *Molecule*.

GetMolecularFormula

```
$FormulaString = $Molecule->GetMolecularFormula(  
    [$IncludeMissingHydrogens,  
    $IncludeNonElements]);
```

Returns molecular formula of a *Molecule* by collecting information about all atoms in the molecule and composing the formula using Hills ordering system:

- o C shows up first and H follows assuming C is present.
- o All other standard elements are sorted alphanumerically.
- o All other non-standard atom symbols are also sorted alphanumerically and follow standard elements.

Notes:

- o By default, missing hydrogens and nonelements are also included.
- o Elements for disconnected fragments are combined into the same formula.
- o Formal charge is also used during composition of molecular formula.

GetMolecularWeight

```
$MolWeight = $Molecule->GetMolecularWeight();
```

Returns molecular weight of a *Molecule* corresponding to sum of atomic weights of all the atoms

GetNumOfAtoms

```
$NumOfAtoms = $Molecule->GetNumOfAtoms();
```

Returns number of atoms in a *Molecule*

GetNumOfBonds

```
$NumOfBonds = $Molecule->GetNumOfBonds();
```

Returns number of bonds in a *Molecule*

GetNumOfConnectedComponents

```
$NumOfComponents = $Molecule->GetNumOfConnectedComponents();
```

Returns number of connected components in a *Molecule*

GetNumOfHeavyAtoms

```
$NumOfHeavyAtoms = $Molecule->GetNumOfHeavyAtoms();
```

Returns number of heavy atoms, non-hydrogen atoms, in a *Molecule*

GetNumOfHydrogenAtoms

```
$NumOfHydrogenAtoms = $Molecule->GetNumOfHydrogenAtoms();
```

Returns number of hydrogen atoms in a *Molecule*

GetNumOfMissingHydrogenAtoms

```
$NumOfMissingHydrogenAtoms = $Molecule->GetNumOfMissingHydrogenAtoms();
```

Returns number of hydrogen atoms in a *Molecule*

GetNumOfNonHydrogenAtoms

```
$NumOfNonHydrogenAtoms = $Molecule->GetNumOfNonHydrogenAtoms();
```

Returns number of non-hydrogen atoms in a *Molecule*.

GetNumOfRings

```
$RingCount = $Molecule->GetNumOfRings();
```

Returns number of rings in a *Molecule*

GetNumOfRingsWithEvenSize

```
$RingCount = $Molecule->GetNumOfRingsWithEvenSize();
```

Returns number of rings with even size in a *Molecule*

GetNumOfRingsWithOddSize

```
$RingCount = $Molecule->GetNumOfRingsWithOddSize();
```

Returns number of rings with odd size in a *Molecule*

GetNumOfRingsWithSize

```
$RingCount = $Molecule->GetNumOfRingsWithSize($Size);
```

Returns number of rings with *Size* in a *Molecule*

GetNumOfRingsWithSizeGreaterThan

```
$RingCount = $Molecule->GetNumOfRingsWithSizeGreaterThan($Size);
```

Returns number of rings with size greater than *Size* in a *Molecule*

GetNumOfRingsWithSizeLessThan

```
$RingCount = $Molecule->GetNumOfRingsWithSizeLessThan($Size);
```

Returns number of rings with size less than *Size* in a *Molecule*

GetRingBonds

```
@RingBonds = $Molecule->GetRingBonds(@RingAtoms);
```

Returns an array of ring Bond objects corresponding to an array of ring *Atoms* in a *Molecule*.

GetRings

```
@Rings = $Molecule->GetRings();
```

Returns rings as an array containing references to arrays of ring *Atom* objects in a *Molecule*

GetRingsWithEvenSize

```
@Rings = $Molecule->GetRingsWithEvenSize();
```

Returns even size rings as an array containing references to arrays of ring *Atom* objects in a *Molecule*

GetRingsWithOddSize

```
@Rings = $Molecule->GetRingsWithOddSize();
```

Returns odd size rings as an array containing references to arrays of ring *Atom* objects in a *Molecule*

GetRingsWithSize

```
@Rings = $Molecule->GetRingsWithSize($Size);
```

Returns rings with *Size* as an array containing references to arrays of ring *Atom* objects in a *Molecule*

GetRingsWithSizeGreaterThan

```
@Rings = $Molecule->GetRingsWithSizeGreaterThan($Size);
```

Returns rings with size greater than *Size* as an array containing references to arrays of ring *Atom* objects in a *Molecule*

GetRingsWithSizeLessThan

```
@Rings = $Molecule->GetRingsWithSizeLessThan($Size);
```

Returns rings with size less than *Size* as an array containing references to arrays of ring *Atom* objects in a *Molecule*

GetSizeOfLargestRing

```
$Size = $Molecule->GetSizeOfLargestRing();
```

Returns size of the largest ring in a *Molecule*

GetSizeOfSmallestRing

```
$Size = $Molecule->GetSizeOfSmallestRing();
```

Returns size of the smallest ring in a *Molecule*

GetSmallestRing

```
@RingAtoms = $Molecule->GetSmallestRing();
```

Returns an array containing *Atom* objects corresponding to the smallest ring in a *Molecule*

GetSpinMultiplicity

```
$SpinMultiplicity = $Molecule->GetSpinMultiplicity();
```

Returns net spin multiplicity of a *Molecule* using one of the following two methods: explicitly set *SpinMultiplicity* property or sum of spin multiplicity on each atom.

GetTopologicallySortedAtoms

```
@SortedAtoms = $Molecule->GetTopologicallySortedAtoms([$StartAtom]);
```

Returns an array of topologically sorted *Atom* objects starting from *StartAtom* or an arbitrary atom in a *Molecule*

HasAtom

```
$Status = $Molecule->HasAtom($Atom);
```

Returns 1 or 0 based on whether *Atom* is present in a *Molecule*

HasBond

```
$Status = $Molecule->HasBond($Bond);
```

Returns 1 or 0 based on whether *Bond* is present in a *Molecule*

HasFusedRings

```
$Status = $Molecule->HasFusedRings();
```

Returns 1 or 0 based on whether any fused rings set is present in a *Molecule*

HasNoRings

```
$Status = $Molecule->HasNoRings();
```

Returns 0 or 1 based on whether any ring is present in a *Molecule*

HasOnlyOneRing

```
$Status = $Molecule->HasOnlyOneRing();
```

Returns 1 or 0 based on whether only one ring is present in a *Molecule*

HasRings

```
$Status = $Molecule->HasRings();
```

Returns 1 or 0 based on whether rings are present in a *Molecule*

IsMolecule

```
$Status = Molecule::IsMolecule();
```

Returns 1 or 0 based on whether *Object* is a *Molecule* object

IsMolecule

```
$Status = $Molecule->IsThreeDimensional();
```

Returns 1 or 0 based on whether any atom in *Molecule* has a non-zero value for Z coordinate

KeepLargestComponent

```
$Molecule->KeepLargestComponent();
```

Deletes atoms corresponding to all other connected components Except for the largest connected component in a *Molecule* and returns *Molecule*

NewAtom

```
$NewAtom = $Molecule->NewAtom(%AtomPropertyNamesAndValues);
```

Creates a new atom using *AtomPropertyNamesAndValues*, add its to *Molecule*, and returns new Atom object

NewBond

```
$Return = $Molecule->NewBond(%BondPropertyNamesAndValues);
```

Creates a new bond using *AtomPropertyNamesAndValues*, add its to *Molecule*, and returns new Bond object

SetActiveRings

```
$Molecule->SetActiveRings($RingsType);
```

Sets up type of detected ring sets to use during all ring related methods and returns *Molecule*. Possible *RingType* values: *Independent* or *All*. By default, *Independent* ring set is used during all ring methods.

StringifyMolecule

```
$MoleculeString = $Molecule->StringifyMolecule();
```

Returns a string containing information about *Molecule* object

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

Atom.pm, Bond.pm, MoleculeFileIO.pm, MolecularFormula.pm

COPYRIGHT

Copyright (C) 2004-2008 Manish Sud. All rights reserved.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser

General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.