## NAME

RDKitUtil

## SYNOPSIS

import RDKitUtil

## DESCRIPTION

RDKitUtil module provides the following functions:

AreAtomIndicesSequentiallyConnected, AreAtomMapNumbersPresentInMol, AreHydrogensMissingInMolecule, CalculateFormalCharge, CalculateSpinMultiplicity, ClearAtomMapNumbers, ConstrainAndEmbed, FilterSubstructureMatchByAtomMapNumbers, FilterSubstructureMatchesByAtomMapNumbers, GenerateBase64EncodedMolStringWithConfIDs, GenerateBase64EncodedMolStrings, GenerateBase64EncodedMolStringsWithIDs, GetAtomIndices, GetAtomMapIndices, GetAtomMapIndicesAndMapNumbers, GetAtomPositions, GetAtomSymbols, GetFormalCharge, GetHeavyAtomNeighbors, GetInlineSVGForMolecule, GetInlineSVGForMolecules, GetMolName, GetNumFragments, GetNumHeavyAtomNeighbors, GetPsi4XYZFormatString, GetSVGForMolecule, GetSVGForMolecules, GetSpinMultiplicity, GetTorsionsAroundRotatableBonds, IsAtomSymbolPresentInMol, IsMolEmpty, IsValidAtomIndex, IsValidElementSymbol, MolFromBase64EncodedMolString, MolFromSubstructureMatch, MolToBase64EncodedMolString, MoleculesWriter, MolsFromSubstructureMatches, ReadAndValidateMolecules, ReadMolecules, ReadMoleculesFromMol2File, ReadMoleculesFromMolFile, ReadMoleculesFromPDBFile, ReadMoleculesFromSDFile, ReadMoleculesFromSMILESFile, ReorderAtomIndicesInSequentiallyConnectedManner, SetAtomPositions, SetWriterMolProps, ValidateElementSymbols, WriteMolecules

## FUNCTIONS

### AreAtomIndicesSequentiallyConnected

```
AreAtomIndicesSequentiallyConnected(Mol, AtomIndices)
```

Check for the presence bonds between sequential pairs of atoms in a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
AtomIndices (list): List of atom indices.
```

*Returns:*

```
bool : True - Sequentially connected; Otherwise, false.
```

### AreAtomMapNumbersPresentInMol

```
AreAtomMapNumbersPresentInMol(Mol)
```

Check for the presence of atom map numbers in a molecue.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
bool : True - Atom map numbers present; Otherwise, false.
```

### AreHydrogensMissingInMolecule

```
AreHydrogensMissingInMolecule(Mol)
```

Check for any missing hydrogens in in a molecue.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
bool : True - Missing hydrogens; Otherwise, false.
```

CalculateFormalCharge

```
CalculateFormalCharge(Mol)
```

Calculate formal charge of a molecule. The formal charge is calculated using RDKit function Chem.GetFormalCharge(Mol).

*Arguments:*

```
Mol (object): RDKit molecule object.
    retrieve formal charge.
```

*Returns:*

```
int : Formal charge.
```

CalculateSpinMultiplicity

```
CalculateSpinMultiplicity(Mol)
```

Calculate spin multiplicity of a molecule. The spin multiplicity is calculated from the number of free radical electrons using Hund's rule of maximum multiplicity defined as 2S + 1 where S is the total electron spin. The total spin is 1/2 the number of free radical electrons in a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
int : Spin multiplicity.
```

ClearAtomMapNumbers

```
ClearAtomMapNumbers(Mol, AllowImplicitValence = True, ClearRadicalElectrons = True)
```

Check and clear atom map numbers in a molecule. In addition, allow implicit valence and clear radical electrons for atoms with associated map numbers.

For example, the following atomic properties are assigned by RDKit to atom map number 1 in a molecule corresponding to SMILES C[C:1](C)C:

NoImplicit: True; ImplicitValence: 0; ExplicitValence: 3; NumExplicitHs: 0; NumImplicitHs: 0; NumRadicalElectrons: 1

This function clears atoms map numbers in the molecule leading to SMILES CC(C)C, along with optionally updating atomic properties as shown below:

NoImplicit: False; ImplicitValence: 1; ExplicitValence: 3; NumExplicitHs: 0; NumImplicitHs: 1; NumRadicalElectrons: 0

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
Mol (object): RDKit molecule object.
```

ConstrainAndEmbed

```
ConstrainAndEmbed(mol, core, coreMatchesMol=None, useTethers=True, coreConfId=-1,
randomseed=2342, getForceField=AllChem.UFFGetMoleculeForceField, **kwargs)
```

The function is a local copy of RDKit fucntion AllChem.ConstrainedEmbed(). It has been enhanced to support an explicit list of core matches corresponding to the matched atom indices in the molecule. The number of matched atom indices must be equal to the number of atoms in core molecule.

*Arguments:*

```
mol (object): RDKit molecule object to embed.
core (object): RDKit molecule to use as a source of constraints.
coreMatchesMol (list): A list matches atom indices in mol.
useTethers: (bool) if True, the final conformation will be optimized
```

```
            subject to a series of extra forces that pull the matching atoms to
            the positions of the core atoms. Otherwise simple distance
            constraints based on the core atoms will be used in the
            optimization.
        coreConfId (int): ID of the core conformation to use.
        randomSeed (int): Seed for the random number generator
```

*Returns:*

```
    mol (object): RDKit molecule object.
```

### FilterSubstructureMatchByAtomMapNumbers

```
    FilterSubstructureMatchByAtomMapNumbers(Mol, PatternMol, AtomIndices)
```

Filter a list of matched atom indices by map atom numbers present in a pattern molecule. The list of atom indices correspond to a list retrieved by RDKit function GetSubstructureMatches using SMILES/SMARTS pattern. The atom map numbers are mapped to appropriate atom indices during the generation of molecules. For example: [O:1]=[S:2](=[O])[C:3][C:4].

*Arguments:*

```
    Mol (object): RDKit molecule object.
    PatternMol (object): RDKit molecule object for a SMILES/SMARTS pattern.
    AtomIndices (list): Atom indices.
```

*Returns:*

```
    list : A list of filtered atom indices.
```

### FilterSubstructureMatchesByAtomMapNumbers

```
    FilterSubstructureMatchesByAtomMapNumbers(Mol, PatternMol, AtomIndicesList)
```

Filter a list of lists containing matched atom indices by map atom numbers present in a pattern molecule. The list of atom indices correspond to a list retrieved by RDKit function GetSubstructureMatches using SMILES/SMARTS pattern. The atom map numbers are mapped to appropriate atom indices during the generation of molecules. For example: [O:1]=[S:2](=[O])[C:3][C:4].

*Arguments:*

```
    Mol (object): RDKit molecule object.
    PatternMol (object): RDKit molecule object for a SMILES/SMARTS pattern.
    AtomIndicesList (list): A list of lists containing atom indices.
```

*Returns:*

```
    list : A list of lists containing filtered atom indices.
```

### GenerateBase64EncodedMolStringWithConfIDs

```
    GenerateBase64EncodedMolStringWithConfIDs(Mol, MolIndex, ConfIDs, PropertyPickleFlags =
Chem.PropertyPickleOptions.AllProps)
```

Setup an iterator generating base64 encoded molecule string for a molecule. The iterator returns a list containing a molecule index, an encoded molecule string, and conf ID.

The molecules are pickled using RDKit Mol.ToBinary() function before their encoding.

*Arguments:*

```
    Mol (object): RDKit molecule object.
    MolIndex (int): Molecule index.
    ConfIDs (list): Conformer IDs.
    PropertyFlags: RDKit property pickle options.
```

*Returns:*

```
    object : Base64 endcoded molecules iterator. The iterator returns a
        list containing a molecule index, an encoded molecule string, and
        conf ID.
```

The following property pickle flags are currently available in RDKit:

```
Chem.PropertyPickleOptions.NoProps
Chem.PropertyPickleOptions.MolProps
Chem.PropertyPickleOptions.AtomProps
Chem.PropertyPickleOptions.BondProps
Chem.PropertyPickleOptions.PrivateProps
Chem.PropertyPickleOptions.AllProps
```

*Example(s):*

```
EncodedMolsInfo = GenerateBase64EncodedMolStringWithConfIDs(Mol, MolIndex, ConfIDs)
for MolIndex, EncodedMol, ConfID in EncodedMolsInfo:
    if EncodeMol is not None:
        Mol = MolFromBase64EncodedMolString(EncodedMol)
```

### GenerateBase64EncodedMolStrings

```
    GenerateBase64EncodedMolStrings(Mols, PropertyPickleFlags =
Chem.PropertyPickleOptions.AllProps)
```

Setup an iterator for generating base64 encoded molecule string from a RDKit molecule iterator. The iterator returns a list containing a molecule index and encoded molecule string or None.

The molecules are pickled using RDKit Mol.ToBinary() function before their encoding.

*Arguments:*

```
iterator: RDKit molecules iterator.
PropertyFlags: RDKit property pickle options.
```

*Returns:*

```
object : Base64 endcoded molecules iterator. The iterator returns a
    list containing a molecule index and an encoded molecule string
    or None.
```

The following property pickle flags are currently available in RDKit:

```
Chem.PropertyPickleOptions.NoProps
Chem.PropertyPickleOptions.MolProps
Chem.PropertyPickleOptions.AtomProps
Chem.PropertyPickleOptions.BondProps
Chem.PropertyPickleOptions.PrivateProps
Chem.PropertyPickleOptions.AllProps
```

*Example(s):*

```
EncodedMolsInfo = GenerateBase64EncodedMolStrings(Mols)
for MolIndex, EncodedMol in EncodedMolsInfo:
    if EncodeMol is not None:
        Mol = MolFromBase64EncodedMolString(EncodedMol)
```

### GenerateBase64EncodedMolStringsWithIDs

```
    GenerateBase64EncodedMolStringsWithIDs(Mols, MolIDs, PropertyPickleFlags =
Chem.PropertyPickleOptions.AllProps)
```

Setup an iterator for generating base64 encoded molecule string from a RDKit molecule iterator. The iterator returns a list containing a molecule ID and encoded molecule string or None.

The molecules are pickled using RDKit Mol.ToBinary() function before their encoding.

*Arguments:*

```
iterator: RDKit molecules iterator.
MolIDs (list): Molecule IDs.
PropertyFlags: RDKit property pickle options.
```

*Returns:*

```
    object : Base64 endcoded molecules iterator. The iterator returns a
        list containing a molecule ID and an encoded molecule string
        or None.
```

The following property pickle flags are currently available in RDKit:

```
Chem.PropertyPickleOptions.NoProps
Chem.PropertyPickleOptions.MolProps
Chem.PropertyPickleOptions.AtomProps
Chem.PropertyPickleOptions.BondProps
Chem.PropertyPickleOptions.PrivateProps
Chem.PropertyPickleOptions.AllProps
```

*Example(s):*

```
EncodedMolsInfo = GenerateBase64EncodedMolStringsWithIDs(Mols)
for MolID, EncodedMol in EncodedMolsInfo:
    if EncodeMol is not None:
        Mol = MolFromBase64EncodedMolString(EncodedMol)
```

### GetAtomIndices

```
GetAtomIndices(Mol)
```

Retrieve a list containing atom indices of all atoms a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
list : List of atom indices.
```

### GetAtomMapIndices

```
GetAtomMapIndices(Mol)
```

Get a list of available atom indices corresponding to atom map numbers present in a SMILES/SMARTS pattern used for creating a molecule. The list of atom indices is sorted in ascending order by atom map numbers.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
list : List of atom indices sorted in the ascending order of atom map
    numbers or None.
```

### GetAtomMapIndicesAndMapNumbers

```
GetAtomMapIndicesAndMapNumbers(Mol)
```

Get lists of available atom indices and atom map numbers present in a SMILES/SMARTS pattern used for creating a molecule. Both lists are sorted in ascending order by atom map numbers.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
list : List of atom indices sorted in the ascending order of atom map
    numbers or None.
list : List of atom map numbers sorted in the ascending order or None.
```

### GetAtomPositions

```
GetAtomPositions(Mol, ConfID = -1)
```

Retrieve a list of lists containing coordinates of all atoms in a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
ConfID (int): Conformer number.
```

*Returns:*

```
list : List of lists containing atom positions.
```

*Example(s):*

```
for AtomPosition in RDKitUtil.GetAtomPositions(Mol):
    print("X: %s; Y: %s; Z: %s" % (AtomPosition[0], AtomPosition[1], AtomPosition[2]))
```

### GetAtomSymbols

```
GetAtomSymbols(Mol)
```

Retrieve a list containing atom symbols of all atoms a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
list : List of atom symbols.
```

### GetFormalCharge

```
GetFormalCharge(Mol, CheckMolProp = True)
```

Get formal charge of a molecule. The formal charge is either retrieved from 'FormalCharge' molecule property or calculated using RDKit function Chem.GetFormalCharge(Mol).

The 'FormalCharge' molecule property may contain multiple space delimited values. The total formal charge corresponds to the sum of the specified formal charge values.

*Arguments:*

```
Mol (object): RDKit molecule object.
CheckMolProp (bool): Check 'FormalCharge' molecule property to
    retrieve formal charge.
```

*Returns:*

```
int : Formal charge.
```

### GetHeavyAtomNeighbors

```
GetHeavyAtomNeighbors(Atom)
```

Get a list of heavy atom neighbors.

*Arguments:*

```
Atom (object): RDKit atom object.
```

*Returns:*

```
list : List of heavy atom neighbors.
```

### GetInlineSVGForMolecule

```
GetInlineSVGForMolecule(Mol, Width, Height, Legend = None, AtomListToHighlight = None,
BondListToHighlight = None, BoldText = True, Base64Encoded = True)
```

Get SVG image text for a molecule suitable for inline embedding into a HTML page.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

```
    Width (int): Width of a molecule image in pixels.
    Height (int): Height of a molecule image in pixels.
    Legend (str): Text to display under the image.
    AtomListToHighlight (list): List of atoms to highlight.
    BondListToHighlight (list): List of bonds to highlight.
    BoldText (bool): Flag to make text bold in the image of molecule.
    Base64Encoded (bool): Flag to return base64 encoded string.
```

*Returns:*

```
    str : SVG image text for inline embedding into a HTML page using "img"
        tag: <img src="data:image/svg+xml;charset=UTF-8,SVGImageText> or
        tag: <img src=">
```

### GetInlineSVGForMolecules

```
    GetInlineSVGForMolecules(Mols, MolsPerRow, MolWidth, MolHeight, Legends = None,
AtomListsToHighlight = None, BondListsToHighLight = None, BoldText = True, Base64Encoded =
True)
```

Get SVG image text for molecules suitable for inline embedding into a HTML page.

*Arguments:*

```
    Mols (list): List of RDKit molecule objects.
    MolsPerRow (int): Number of molecules per row.
    Width (int): Width of a molecule image in pixels.
    Height (int): Height of a molecule image in pixels.
    Legends (list): List containing strings to display under images.
    AtomListsToHighlight (list): List of lists containing atoms to highlight
        for molecules.
    BondListsToHighlight (list): List of lists containing bonds to highlight
        for molecules
    BoldText (bool): Flag to make text bold in the image of molecules.
    Base64Encoded (bool): Flag to return base64 encoded string.
```

*Returns:*

```
    str : SVG image text for inline embedding into a HTML page using "img"
        tag: <img src="data:image/svg+xml;charset=UTF-8,SVGImageText> or
        tag: <img src=">
```

### GetMolName

```
    GetMolName(Mol, MolNum = None)
```

Get molecule name.

*Arguments:*

```
    Mol (object): RDKit molecule object.
    MolNum (int or None): Molecule number in input file.
```

*Returns:*

```
    str : Molname corresponding to _Name property of a molecule, generated
        from specieid MolNum using the format "Mol%d" % MolNum, or an
        empty string.
```

### GetNumFragments

```
    GetNumFragments(Mol)
```

Get number of fragment in a molecule.

*Arguments:*

```
    Atom (object): RDKit molecule object.
```

*Returns:*

```
    int : Number of fragments.
```

GetNumHeavyAtomNeighbors

```
    GetNumHeavyAtomNeighbors(Atom)
```

Get number of heavy atom neighbors.

*Arguments:*

```
    Atom (object): RDKit atom object.
```

*Returns:*

```
    int : Number of neighbors.
```

GetPsi4XYZFormatString

```
    GetPsi4XYZFormatString(Mol, ConfID = -1, FormalCharge = "auto", SpinMultiplicity =
"auto", Symmetry = "auto", NoCom = False, NoReorient = False, CheckFragments = False)
```

Retrieve geometry string of a molecule in Psi4ish XYZ format to perform Psi4 quantum chemistry calculations.

You may explicit specify multiple space delimited values for formal charge and spin multiplicity. Otherwise, these values are either automatically retrieved from 'FormalCharge' and 'SpinMultiplicity' molecule properties or calculated using RDKit. The number of specified values for these properties must match the number of fragments in the molecule during the processing of the fragments.

*Arguments:*

```
    Mol (object): RDKit molecule object.
    ConfID (int): Conformer number.
    FormalCharge (str): Specified formal charge or 'auto' to calculate
       its value by RDKit.
    SpinMultiplicity (str): Specified spin multiplicity or 'auto' to calculate
       its value by RDKit.
    Symmetry (str): Specified symmetry or 'auto' to calculate its value by
       Psi4.
    NoCom (bool): Flag to disable recentering of a molecule by Psi4.
    NoReorient (bool): Flag to disable reorientation of a molecule by Psi4.
    CheckFragments (bool): Check for fragments and setup geometry string
       using  -- separator between fragments.
```

*Returns:*

```
    str : Geometry string of a molecule in Psi4ish XYZ format.
```

GetSVGForMolecule

```
    GetSVGForMolecule(Mol, Width, Height, Legend = None, AtomListToHighlight = None,
BondListToHighlight = None, BoldText = True)
```

Get SVG image text for a molecule suitable for viewing in a browser.

*Arguments:*

```
    Mol (object): RDKit molecule object.
    Width (int): Width of a molecule image in pixels.
    Height (int): Height of a molecule image in pixels.
    Legend (str): Text to display under the image.
    AtomListToHighlight (list): List of atoms to highlight.
    BondListToHighlight (list): List of bonds to highlight.
    BoldText (bool): Flag to make text bold in the image of molecule.
```

*Returns:*

```
    str : SVG image text for writing to a SVG file for viewing in a browser.
```

GetSVGForMolecules

```
    GetSVGForMolecules(Mols, MolsPerRow, MolWidth, MolHeight, Legends = None,
AtomListsToHighlight = None, BondListsToHighlight = None, BoldText = True)
```

Get SVG image text for molecules suitable for viewing in a browser.

*Arguments:*

```
    Mols (list): List of RDKit molecule objects.
    MolsPerRow (int): Number of molecules per row.
    Width (int): Width of a molecule image in pixels.
    Height (int): Height of a molecule image in pixels.
    Legends (list): List containing strings to display under images.
    AtomListsToHighlight (list): List of lists containing atoms to highlight
        for molecules.
    BondListsToHighlight (list): List of lists containing bonds to highlight
        for molecules
    BoldText (bool): Flag to make text bold in the image of molecules.
```

*Returns:*

```
    str : SVG image text for writing to a SVG file for viewing in a browser.
```

### GetSpinMultiplicity

```
    GetSpinMultiplicity(Mol, CheckMolProp = True)
```

Get spin multiplicity of a molecule. The spin multiplicity is either retrieved from 'SpinMultiplicity' molecule property or calculated from the number of free radical electrons using Hund's rule of maximum multiplicity defined as 2S + 1 where S is the total electron spin. The total spin is 1/2 the number of free radical electrons in a molecule.

The 'SpinMultiplicity' molecule property may contain multiple space delimited values. The total spin multiplicity corresponds to the total number of free radical electrons which are calculated for each specified value.

*Arguments:*

```
    Mol (object): RDKit molecule object.
    CheckMolProp (bool): Check 'SpinMultiplicity' molecule property to
        retrieve spin multiplicity.
```

*Returns:*

```
    int : Spin multiplicity.
```

### GetTorsionsAroundRotatableBonds

```
    GetTorsionsAroundRotatableBonds(Mol, RotBondsPatternMol, IgnoreHydrogens = True)
```

Identify torsions around rotatable bonds and return a list of lists containing atom indices of torsions.

*Arguments:*

```
    Mol (object): RDKit molecule object.
    PatternMol (object): RDKit molecule object corresponding to SMARTS
        pattern to identify rotatable bonds.
    IgnoreHydrogens (bool): Flag to include torsions around rotatable bonds
        containing hydrogens.
```

*Returns:*

```
    list : List of lists containing atom indices of torsions around
        rotatable bonds.
```

### IsAtomSymbolPresentInMol

```
    IsAtomSymbolPresentInMol(Mol, AtomSymbol, IgnoreCase = True)
```

Check for the presence of an atom symbol in a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
AtomSymbol (str): Atom symbol.
```

*Returns:*

```
bool : True - Atom symbol in molecule; Otherwise, false.
```

IsMolEmpty

```
IsMolEmpty(Mol)
```

Check for the presence of atoms in a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
```

*Returns:*

```
bool : True - No atoms in molecule; Otherwise, false.
```

IsValidAtomIndex

```
IsValidAtomIndex(Mol, AtomIndex)
```

Validate presence atom index in a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
AtomIndex (int): Atom index.
```

*Returns:*

```
bool : True - Valid atom index; Otherwise, false.
```

IsValidElementSymbol

```
IsValidElementSymbol(ElementSymbol)
```

Validate element symbol.

*Arguments:*

```
ElementSymbol (str): Element symbol
```

*Returns:*

```
bool : True - Valid element symbol; Otherwise, false.
```

MolFromBase64EncodedMolString

```
MolFromBase64EncodedMolString(EncodedMol)
```

Generate a RDKit molecule object from a base64 encoded string.

*Arguments:*

```
str: Base64 encoded molecule string.
```

*Returns:*

```
object : RDKit molecule object or None.
```

MolFromSubstructureMatch

```
MolFromSubstructureMatch(Mol, PatternMol, AtomIndices, FilterByAtomMapNums = False)
```

Generate a RDKit molecule object for a list of matched atom indices present in a pattern molecule. The list of atom indices correspond to a list retrieved by RDKit function GetSubstructureMatches using SMILES/SMARTS pattern. The atom indices are optionally filtered by mapping atom numbers to appropriate

atom indices during the generation of the molecule. For example: [O:1]=[S:2](=[O])[C:3][C:4].

*Arguments:*

```
Mol (object): RDKit molecule object.
PatternMol (object): RDKit molecule object for a SMILES/SMARTS pattern.
AtomIndices (list): Atom indices.
FilterByAtomMapNums (bool): Filter matches by atom map numbers.
```

*Returns:*

```
object : RDKit molecule object or None.
```

MolToBase64EncodedMolString

```
MolToBase64EncodedMolString(Mol, PropertyPickleFlags =
Chem.PropertyPickleOptions.AllProps)
```

Encode RDkit molecule object into a base64 encoded string. The properties can be optionally excluded.

The molecule is pickled using RDKit Mol.ToBinary() function before their encoding.

*Arguments:*

```
Mol (object): RDKit molecule object.
PropertyPickleFlags: RDKit property pickle options.
```

*Returns:*

```
str : Base64 encode molecule string or None.
```

The following property pickle flags are currently available in RDKit:

```
Chem.PropertyPickleOptions.NoProps
Chem.PropertyPickleOptions.MolProps
Chem.PropertyPickleOptions.AtomProps
Chem.PropertyPickleOptions.BondProps
Chem.PropertyPickleOptions.PrivateProps
Chem.PropertyPickleOptions.AllProps
```

MoleculesWriter

```
MoleculesWriter(FileName, **KeyWordArgs)
```

Set up a molecule writer.

*Arguments:*

```
FileName (str): Name of a file with complete path.
**KeyWordArgs (dictionary) : Parameter name and value pairs for writing and
    processing molecules.
```

*Returns:*

```
RDKit object : Molecule writer.
```

The file extension is used to determine type of the file and set up an appropriate file writer.

MolsFromSubstructureMatches

```
MolsFromSubstructureMatches(Mol, PatternMol, AtomIndicesList, FilterByAtomMapNums =
False)
```

Generate a list of RDKit molecule objects for a list containing lists of matched atom indices present in a pattern molecule. The list of atom indices correspond to a list retrieved by RDKit function GetSubstructureMatches using SMILES/SMARTS pattern. The atom indices are optionally filtered by mapping atom numbers to appropriate atom indices during the generation of the molecule. For example: [O:1]=[S:2](=[O])[C:3][C:4].

*Arguments:*

```
Mol (object): RDKit molecule object.
PatternMol (object): RDKit molecule object for a SMILES/SMARTS pattern.
```

```
    AtomIndicesList (list): A list of lists containing atom indices.
    FilterByAtomMapNums (bool): Filter matches by atom map numbers.
```

*Returns:*

```
    list : A list of lists containg RDKit molecule objects or None.
```

ReadAndValidateMolecules

```
    ReadAndValidateMolecules(FileName, **KeyWordArgs)
```

Read molecules from an input file, validate all molecule objects, and return a list of valid and non-valid molecule objects along with their counts.

*Arguments:*

```
    FileName (str): Name of a file with complete path.
    **KeyWordArgs (dictionary) : Parameter name and value pairs for reading and
        processing molecules.
```

*Returns:*

```
    list : List of valid RDKit molecule objects.
    int : Number of total molecules in input file.
    int : Number of valid molecules in input file.
```

The file extension is used to determine type of the file and set up an appropriate file reader.

ReadMolecules

```
    ReadMolecules(FileName, **KeyWordArgs)
```

Read molecules from an input file without performing any validation and creation of molecule objects.

*Arguments:*

```
    FileName (str): Name of a file with complete path.
    **KeyWordArgs (dictionary) : Parameter name and value pairs for reading and
        processing molecules.
```

*Returns:*

```
    list : List of RDKit molecule objects.
```

The file extension is used to determine type of the file and set up an appropriate file reader.

ReadMoleculesFromMol2File

```
    ReadMoleculesFromMol2File(FileName, Sanitize = True, RemoveHydrogens = True)
```

Read molecules from a Tripos Mol2 file. The first call to the function creates and returns a generator object using Python yield statement. The molecules are created during the subsequent iteration by the generator object.

*Arguments:*

```
    FileName (str): Name of a file with complete path.
    Sanitize (bool): Sanitize molecules.
    RemoveHydrogens (bool): Remove hydrogens from molecules.
```

*Returns:*

```
    list : A Python generator object for iterating over the molecules.
```

ReadMoleculesFromMolFile

```
    ReadMoleculesFromMolFile(FileName, Sanitize = True, RemoveHydrogens = True,
StrictParsing = True)
```

Read molecule from a MDL Mol file.

*Arguments:*

```
    FileName (str): Name of a file with complete path.
    Sanitize (bool): Sanitize molecules.
    RemoveHydrogens (bool): Remove hydrogens from molecules.
    StrictParsing (bool): Perform strict parsing.
```

*Returns:*

```
    list : List of RDKit molecule objects.
```

ReadMoleculesFromPDBFile

```
    ReadMoleculesFromPDBFile(FileName, Sanitize = True, RemoveHydrogens = True)
```

Read molecule from a PDB file.

*Arguments:*

```
    FileName (str): Name of a file with complete path.
    Sanitize (bool): Sanitize molecules.
    RemoveHydrogens (bool): Remove hydrogens from molecules.
```

*Returns:*

```
    list : List of RDKit molecule objects.
```

ReadMoleculesFromSDFile

```
    ReadMoleculesFromSDFile(FileName, Sanitize = True, RemoveHydrogens = True,
StrictParsing = True)
```

Read molecules from a SD file.

*Arguments:*

```
    FileName (str): Name of a file with complete path.
    Sanitize (bool): Sanitize molecules.
    RemoveHydrogens (bool): Remove hydrogens from molecules.
    StrictParsing (bool): Perform strict parsing.
```

*Returns:*

```
    list : List of RDKit molecule objects.
```

ReadMoleculesFromSMILESFile

```
    ReadMoleculesFromSMILESFile(FileName, SMILESDelimiter = ' ', SMILESColIndex = 0,
SMILESNameColIndex = 1, SMILESTitleLine = 1, Sanitize = 1)
```

Read molecules from a SMILES file.

*Arguments:*

```
    SMILESDelimiter (str): Delimiter for parsing SMILES line
    SMILESColIndex (int): Column index containing SMILES string.
    SMILESNameColIndex (int): Column index containing molecule name.
    SMILESTitleLine (int): Flag to indicate presence of title line.
    Sanitize (int): Sanitize molecules.
```

*Returns:*

```
    list : List of RDKit molecule objects.
```

ReorderAtomIndicesInSequentiallyConnectedManner

```
    ReorderAtomIndicesInSequentiallyConnectedManner(Mol, AtomIndices)
```

Check for the presence of sequentially connected list of atoms in an arbitray list of atoms in molecule.

*Arguments:*

```
    Mol (object): RDKit molecule object.
    AtomIndices (list): List of atom indices.
```

*Returns:*

```
bool : True - Sequentially connected list found; Otherwise, false.
list : List of seqeuntially connected atoms or None.
```

SetAtomPositions

```
SetAtomPositions(Mol, AtomPositions, ConfID = -1)
```

Set atom positions of all atoms in a molecule.

*Arguments:*

```
Mol (object): RDKit molecule object.
AtomPositions (object): List of lists containing atom positions.
ConfID (int): Conformer number.
```

*Returns:*

```
object : RDKit molecule object.
```

SetWriterMolProps

```
SetWriterMolProps(Writer, Mol)
```

Setup molecule properties for a writer to output.

*Arguments:*

```
Writer (object): RDKit writer object.
Mol (object): RDKit molecule object.
```

*Returns:*

```
object : Writer object.
```

ValidateElementSymbols

```
ValidateElementSymbols(ElementSymbols)
```

Validate element symbols.

*Arguments:*

```
ElementSymbols (list): List of element symbols to validate.
```

*Returns:*

```
bool : True - All element symbols are valid; Otherwise, false.
```

WriteMolecules

```
WriteMolecules(FileName, Mols, **KeyWordArgs)
```

Write molecules to an output file.

*Arguments:*

```
FileName (str): Name of a file with complete path.
Mols (list): List of RDKit molecule objects.
**KeyWordArgs (dictionary) : Parameter name and value pairs for writing and
    processing molecules.
```

*Returns:*

```
int : Number of total molecules.
int : Number of processed molecules written to output file.
```

The file extension is used to determine type of the file and set up an appropriate file writer.

## AUTHOR

Manish Sud <msud@san.rr.com>

## COPYRIGHT