NAME

RDKitUtil

SYNOPSIS

import RDKitUtil

DESCRIPTION

RDKitUtil module provides the following functions:

AreAtomIndicesSequentiallyConnected, AreAtomMapNumbersPresentInMol, FilterSubstructureMatchByAtomMapNumbers, FilterSubstructureMatchesByAtomMapNumbers, GenerateBase64EncodedMolStrings, GetInlineSVGForMolecule, GetInlineSVGForMolecules, GetMolName, GetSVGForMolecule, GetSVGForMolecules, IsMolEmpty, IsValidElementSymbol, MolFromBase64EncodedMolString, MolFromSubstructureMatch, MolToBase64EncodedMolString, MoleculesWriter, MolsFromSubstructureMatches, ReadAndValidateMolecules, ReadMolecules, ReadMoleculesFromMol2File, ReadMoleculesFromMolFile, ReadMoleculesFromPDBFile, ReadMoleculesFromSDFile, ReadMoleculesFromSMILESFile, SetWriterMolProps, 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.
```

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 comtaining 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.
```

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)
```

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.
```

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.
```

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.
```

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 GetSubstructureMatche 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(s):*

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(s):*

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 molecule from a Tripos Mol2 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.
```

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.
```

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.
```

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

The functionality available in this file is implemented using RDKit, an open source toolkit for cheminformatics

developed by Greg Landrum.

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.