NAME

MiscUtil

SYNOPSIS

import MiscUtil

DESCRIPTION

MiscUtil module provides the following functions:

CheckFileExt, CheckTextValue, DoesSMILESFileContainTitleLine, ExpandFileNames, GetExamplesTextFromDocOptText, GetExcelStyleColumnLabel, GetFormattedElapsedTime, GetFormattedFileSize, GetMayaChemToolsLibDataPath, GetMayaChemToolsVersion, GetTextLines, GetTextLinesWords, GetWallClockAndProcessorTime, IsEmpty, IsFloat, IsInteger, IsNumber, JoinWords, ObjectFromBase64EncodedString, ObjectToBase64EncodedString, ParseFileName, PrintError, PrintInfo, PrintWarning, ProcessOptionConformerGenerator, ProcessOptionConformerParameters, ProcessOptionInfileParameters, ProcessOptionMultiprocessingParameters, ProcessOptionNameValuePairParameters, ProcessOptionOutfileParameters, ProcessOptionPyMOLCubeFileViewParameters, ProcessOptionSeabornPlotParameters, ReplaceHTMLEntitiesInText, TruncateText, ValidateOptionFileExt, ValidateOptionFilePath, ValidateOptionFloatValue, ValidateOptionIntegerValue, ValidateOptionNumberValue, ValidateOptionNumberValues, ValidateOptionTextValue, ValidateOptionsDistinctFileNames, ValidateOptionsOutputFileOverwrite, WrapText

FUNCTIONS

CheckFileExt

```
CheckFileExt(FileName, FileExts)
```

Check file type based on the specified file extensions delimited by spaces.

Arguments:

```
FileName (str): Name of a file.
FileExts (str): Space delimited string containing valid file extensions.
```

Returns:

```
bool : True, FileName contains a valid file extension; Otherwise, False.
```

CheckTextValue

```
CheckTextValue(Value, ValidValues)
```

Check text value based on the specified valid values delimited by spaces.

Arguments:

```
Value (str): Text value
ValidValues (str): Space delimited string containing valid values.
```

Returns:

```
bool : True, Value is valid; Otherwise, False.
```

DoesSMILESFileContainTitleLine

```
DoesSMILESFileContainTitleLine(FileName)
```

Determine whether the SMILES file contain a title line based on the presence of a string SMILES, Name or ID in the first line.

Arguments:

```
FileName (str): Name of a file.
```

Returns:

```
bool : True, File contains title line; Otherwise, False.
```

ExpandFileNames

```
ExpandFileNames(FilesSpec, Delimiter = ",")
```

Expand files specification using glob module to process any * or ? wild cards in file names and return a

list of expanded file names.

Arguments:

```
FilesSpec (str): Files specifications
Delimiter (str): Delimiter for file specifications
```

Returns:

```
list : List of expanded file names
```

GetExamplesTextFromDocOptText

```
GetExamplesTextFromDocOptText(DocOptText)
```

Get script usage example lines from a docopt doc string. The example text line start from a line containing `Examples:` keyword at the beginning of the line.

Arguments:

```
DocOptText (str): Doc string containing script usage examples lines starting with
    a line marked by 'Examples:' keyword at the beginning of a line.
```

Returns:

```
str : A string containing text lines retrieved from the examples section of
    DocOptText parameter.
```

GetExcelStyleColumnLabel

```
GetExcelStyleColumnLabel(ColNum)
```

Return Excel style column label for a colum number.

Arguments:

```
ColNum (int): Column number
```

Returns:

```
str : Excel style column label.
```

GetFormattedElapsedTime

```
GetFormattedElapsedTime(StartingWallClockTime, StartingProcessorTime)
```

Get elapsed wallclock and processor times as a string in the following format: Wallclock: %s days, %d hrs, %d mins, %d secs (Process: %d days, %d hrs, %d mins, %.2f secs).

Arguments:

```
StartingWallClockTime (float): Starting wallclock time in seconds.
StartingProcessorTime (float): Starting processor time in seconds.
```

Returns:

```
str : Elapsed time formatted as:
    Wallclock: %s days, %d hrs, %d mins, %d secs (Process:  %d days,
    %d hrs, %d mins, %.2f secs)
```

GetFormattedFileSize

```
GetFormattedFileSize(FileName, Precision = 1)
```

Get file size as a string in the following format: %.*f <bytes, KB, MB, GB>

Arguments:

```
FileName (str): File path.
Precision (int): File size precision.
```

Returns:

```
str : File size formatted as: %.2f <bytes, KB, MB, GB>
```

GetMayaChemToolsLibDataPath

```
GetMayaChemToolsLibDataPath()
```

Get location of MayaChemTools lib data directory.

Returns:

```
str : Location of MayaChemTools lib data directory.
```

The location of MayaChemTools lib data directory is determined relative to MayaChemTools python lib directory name available through sys.path.

GetMayaChemToolsVersion

```
GetMayaChemToolsVersion()
```

Get version number for MayaChemTools from PackageInfo.csv file in MayaChemTool lib data directory.

Returns:

```
str : Version number
```

GetTextLines

```
GetTextLines(TextFilePath)
```

Read text lines from input file, remove new line characters and return a list containing stripped lines.

Arguments:

```
TextFilePath (str): Text file name including file path.
```

Returns:

```
list : A list lines.
```

GetTextLinesWords

```
GetTextLinesWords(TextFilePath, Delimiter, QuoteChar, IgnoreHeaderLine)
```

Parse lines in the specified text file into words in a line and return a list containing list of parsed line words.

Arguments:

```
TextFilePath (str): Text file name including file path.
Delimiter (str): Delimiter for parsing text lines.
QuoteChar (str): Quote character for line words.
IgnoreHeaderLine (bool): A flag indicating whether to ignore first
    valid data line corresponding to header line.
```

Returns:

```
list : A list of lists containing parsed words for lines.
```

The lines starting with # or // are considered comment lines and are ignored during parsing along with any empty lines.

GetWallClockAndProcessorTime

```
GetWallClockAndProcessorTime()
```

Get wallclock and processor times in seconds.

Returns:

```
float : Wallclock time.
float : Processor time.
```

IsEmpty

```
IsEmpty(Value)
```

Determine whether the specified value is empty after converting it in to a string and removing all leading and trailing white spaces. A value of type None is considered empty.

Arguments:

```
Value (str, int or float): Text or a value
```

Returns:

```
bool : True, Text string is empty; Otherwsie, False.
```

IsFloat

```
IsFloat(Value)
```

Determine whether the specified value is a float by converting it into a float.

Arguments:

```
Value (str, int or float): Text
```

Returns:

```
bool : True, Value is a float; Otherwsie, False.
```

IsInteger

```
IsInteger(Value)
```

Determine whether the specified value is an integer by converting it into an int.

Arguments:

```
Value (str, int or float): Text
```

Returns:

```
bool : True, Value is an integer; Otherwsie, False.
```

IsNumber

```
IsNumber(Value)
```

Determine whether the specified value is a number by converting it into a float.

Arguments:

```
Value (str, int or float): Text
```

Returns:

```
bool : True, Value is a number; Otherwsie, False.
```

JoinWords

```
JoinWords(Words, Delimiter, Quote = False)
```

Join words in a list using specified delimiter with optional quotes around words.

Arguments:

```
Words (list): List containing words to join.
Delimiter (string): Delimiter for joining words.
Quote (bool): Put quotes around words.
```

Returns:

```
str : String containing joined words.
```

ObjectFromBase64EncodedString

```
ObjectFromBase64EncodedString(EncodedObject)
```

Generate Python object from a bas64 encoded and pickled object string.

Arguments:

```
str: Base64 encoded and pickled object string.
```

Returns:

```
object : Python object or None.
```

ObjectToBase64EncodedString

```
ObjectToBase64EncodedString(Object)
```

Encode Python object into base64 encoded string. The object is pickled before encoding.

Arguments:

```
    object: Python object.
```

Returns:

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

ParseFileName

```
    ParseFileName(FilePath)
```

Parse specified file path and return file dir, file name, and file extension.

Arguments:

```
    FilePath (str): Name of a file with complete file path.
```

Returns:

```
    str : File directory.
    str : File name without file extension.
    str : File extension.
```

PrintError

```
    PrintError(Msg, Status=1)
```

Print message to stderr along with flushing stderr and exit with a specified status. An `Error` prefix is placed before the message.

Arguments:

```
    Msg (str): Text message.
    Status (int): Exit status.
```

PrintInfo

```
    PrintInfo(Msg='')
```

Print message to stderr along with flushing stderr.

Arguments:

```
    Msg (str): Text message.
```

PrintWarning

```
    PrintWarning(msg)
```

Print message to stderr along with flushing stderr. An `Warning` prefix is placed before the message.

Arguments:

```
    Msg (str): Text message.
```

ProcessOptionConformerGenerator

```
    ProcessOptionConformerGenerator(OptionName, OptionValue)
```

Process conformer generator option and return a map containing paramater name and values for generatiing conformers.

Arguments:

```
    ParamOptionName (str): Command line conformer generator option name
    ParamOptionValue (str): Command line conformer generator option value
```

Returns:

```
    dictionary: Conformer generation parameter name and value pairs.
```

ProcessOptionConformerParameters

```
    ProcessOptionConformerParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo = None)
```

Process parameters for conformer generation and return a map containing processed parameter names and values.

Arguments:

```
ParamsOptionName (str): Command line conformer generation parameters
    option name.
ParamsOptionValue (str): Comma delimited list of parameter name and value pairs.
ParamsDefaultInfo (dict): Default values to override for selected parameters.
```

Returns:

```
dictionary: Processed parameter name and value pairs.
```

The parameter name and values specified in ParamsOptionValues are validated before returning them in a dictionary.

ProcessOptionInfileParameters

```
ProcessOptionInfileParameters(ParamsOptionName, ParamsOptionValue, InfileName =
None, OutfileName = None, ParamsDefaultInfo = None)
```

Process parameters for reading input files and return a map containing processed parameter names and values.

Arguments:

```
ParamsOptionName (str): Command line input parameters option name.
ParamsOptionValue (str): Comma delimited list of parameter name and value pairs.
InfileName (str): Name of input file.
OutfileName (str): Name of output file.
ParamsDefaultInfo (dict): Default values to override for selected parameters.
```

Returns:

```
dictionary: Processed parameter name and value pairs.
```

The parameter name and values specified in ParamsOptionValue are validated before returning them in a dictionary.

ProcessOptionMultiprocessingParameters

```
ProcessOptionMultiprocessingParameters(ParamsOptionName, ParamsOptionValue)
```

Process parameters for multiprocessing and return a map containing processed parameter names and values.

Arguments:

```
ParamsOptionName (str): Command line multiprocessing parameters option name.
ParamsOptionValue (str): Comma delimited list of parameter name and value pairs.
```

Returns:

```
dictionary: Processed parameter name and value pairs.
```

The parameter name and values specified in ParamsOptionValue are validated before returning them in a dictionary.

ProcessOptionNameValuePairParameters

```
ProcessOptionNameValuePairParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo)
```

Process name and value parameter pairs for an option and return a map containing processed parameter names and values.

Arguments:

```
ParamsOptionName (str): Command line option name for name and value
    parameter pairs.
ParamsOptionValue (str): Comma delimited list of parameter name and
    value parameter pairs.
ParamsDefaultInfo (dict): A dictionary containing a list of parameter
    type and default value pairs keyed by parameter name. Supported
    parameter types: bool, int, float, file, and str.
```

Returns:

```
dictionary: Processed parameter name and value pairs.
```

The parameter names and values specified in ParamsOptionValue are validated before returning them in a dictionary.

Example(s):

```
ParamsDefaultInfo = {"Cleanup": ["bool", True], "RemoveFragments":
    ["bool", True], "Neutralize": ["bool", True],
    "CanonicalizeTautomer": ["bool", True]}
ProcessOptionNameValuePairParameters("--methodologyParams",
    Options["--methodologyParams"], ParamsDefaultInfo)
```

ProcessOptionOutfileParameters

```
ProcessOptionOutfileParameters(ParamsOptionName, ParamsOptionValue, InfileName =
None, OutfileName = None, ParamsDefaultInfo = None)
```

Process parameters for writing output files and return a map containing processed parameter names and values.

Arguments:

```
ParamsOptionName (str): Command line input parameters option name.
ParamsOptionValue (str): Comma delimited list of parameter name and value pairs.
InfileName (str): Name of input file.
OutfileName (str): Name of output file.
ParamsDefaultInfo (dict): Default values to override for selected parameters.
```

Returns:

```
dictionary: Processed parameter name and value pairs.
```

The parameter name and values specified in ParamsOptionValue are validated before returning them in a dictionary.

The default value of some parameters may depend on type of input file. Consequently, the input file name is also needed.

ProcessOptionPyMOLCubeFileViewParameters

```
ProcessOptionPyMOLCubeFileViewParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo = None)
```

Process PyMOl parameters for cube file views and return a map containing processed parameter names and values.

ParamsOptionValue is a comma delimited list of parameter name and value pairs for setting up PyMOL views.

The supported parameter names along with their default and possible values are shown below:

```
ContourColor1, red, ContourColor2, blue,
ContourLevel1, -0.02, ContourLevel2, 0.02,
ContourLevel, 0.02,
ContourLevelAutoAt, 0.5,
ESPRampValues, -1.0 0 1.0,
ESPRampColors, red white blue,
HideHydrogens, yes, DisplayESP, OnSurface,
DisplayMolecule, BallAndStick,
DisplaySphereScale, 0.3, DisplayStickRadius, 0.2,
MeshQuality,2, MeshWidth, 0.5,
SurfaceQualuty, 2, SurfaceTransparency, 0.25,
VolumeColorRamp, auto, VolumeColorRampOpacity, 0.2,
VolumeContourWindowFactor, 0.05
```

Arguments:

```
ParamsOptionName (str): Command line PyMOL view option name.
ParamsOptionValues (str): Comma delimited list of parameter name and value pairs.
ParamsDefaultInfo (dict): Default values to override for selected parameters.
```

Returns:

```
dictionary: Processed parameter name and value pairs.
```

ProcessOptionSeabornPlotParameters

```
    ProcessOptionSeabornPlotParameters(ParamsOptionName, ParamsOptionValue,
ParamsDefaultInfo = None)
```

Process parameters for generating Seaborn plots and return a map containing processed parameter names and values.

Arguments:

```
    ParamsOptionName (str): Command line seaborn parameters option name.
    ParamsOptionValue (str): Comma delimited list of parameter name and value pairs.
    ParamsDefaultValues (dict): Default values for selected parameters.
```

Returns:

```
    dictionary: Processed parameter name and value pairs.
```

The parameter name and values specified in ParamsOptionValue are validated before returning them in a dictionary.

ReplaceHTMLEntitiesInText

```
    ReplaceHTMLEntitiesInText(Text)
```

Check and replace the followng HTML entities to their respective code for display in a browser: < (less than), > (greater than), & (ampersand), " (double quote), and ' (single quote).

Arguments:

```
    Text (str): Text value.
```

Returns:

```
    str : Modifed text value.
```

TruncateText

```
    TruncateText(Text, Width, TrailingChars = "...")
```

Truncate text using specified width along with appending any trailing characters.

Arguments:

```
    Text (string): Input text.
    Width (int): Max number of characters before truncating text.
    Delimiter (string): Trailing characters to append or None.
```

Returns:

```
    str : Truncated text
```

ValidateOptionFileExt

```
    ValidateOptionFileExt(OptionName, FileName, FileExts)
```

Validate file type based on the specified file extensions delimited by spaces.

Arguments:

```
    OptionName (str): Command line option name.
    FileName (str): Name of a file.
    FileExts (str): Space delimited string containing valid file extensions.
```

The function exits with an error message for a file name containing invalid file extension.

ValidateOptionFilePath

```
    ValidateOptionFilePath(OptionName, FilePath)
```

Validate presence of the file.

Arguments:

```
    OptionName (str): Command line option name.
    FilePath (str): Name of a file with complete path.
```

The function exits with an error message for a file path that doesn't exist.

ValidateOptionFloatValue

```
ValidateOptionFloatValue(OptionName, OptionValue, CmpOpValueMap)
```

Validate option value using comparison operater and value pairs in specified in a map.

Arguments:

```
OptionName (str): Command line option name.
OptionValue (float or str): Command line option value.
CmpOpValueMap (dictionary): Comparison operator key and value pairs to
    validate values specified in OptionValue.
```

The function exits with an error message for an invalid option values specified in OptionValue.

Example(s):

```
ValidateOptionNumberValue("-b, --butinaSimilarityCutoff",
    Options["--butinaSimilarityCutoff"],
    {">": 0.0, "<=" : 1.0})
```

ValidateOptionIntegerValue

```
ValidateOptionIntegerValue(OptionName, OptionValue, CmpOpValueMap)
```

Validate option value using comparison operater and value pairs in specified in a map.

Arguments:

```
OptionName (str): Command line option name.
OptionValue (int or str): Command line option value.
CmpOpValueMap (dictionary): Comparison operator key and value pairs to
    validate values specified in OptionValue.
```

The function exits with an error message for an invalid option values specified in OptionValue.

Example(s):

```
ValidateOptionIntegerValue("--maxConfs", Options["--maxConfs"],
    {">": 0})
```

ValidateOptionNumberValue

```
ValidateOptionNumberValue(OptionName, OptionValue, CmpOpValueMap)
```

Validate option value using comparison operater and value pairs in specified in a map.

Arguments:

```
OptionName (str): Command line option name.
OptionValue (int or float): Command line option value.
CmpOpValueMap (dictionary): Comparison operator key and value pairs to
    validate values specified in OptionValue.
```

The function exits with an error message for an invalid option values specified in OptionValue.

Example(s):

```
ValidateOptionNumberValue("--maxConfs", int(Options["--maxConfs"]),
    {">": 0})
ValidateOptionNumberValue("-b, --butinaSimilarityCutoff",
    float(Options["--butinaSimilarityCutoff"]),
    {">": 0.0, "<=" : 1.0})
```

ValidateOptionNumberValues

```
    ValidateOptionNumberValues(OptionName, OptionValueString, OptionValueCount,
OptionValueDelimiter, OptionValueType, CmpOpValueMap)
```

Validate numerical option values using option value string, delimiter, value type, and a specified map containing comparison operator and value pairs.

Arguments:

```
OptionName (str): Command line option name.
OptionValueString (str): Command line option value.
OptionValueCount (int): Number of values in OptionValueString.
OptionValueDelimiter (str): Delimiter used for values in OptionValueString.
OptionValueType (str): Valid number types (integer or float)
CmpOpValueMap (dictionary): Comparison operator key and value pairs to
```

```
validate values specified in OptionValueString.
```

The function exits with an error message for invalid option values specified in OptionValueString

Example(s):

```
ValidateOptionNumberValues("-m, --molImageSize",
    Options["--molImageSize"], 2, ",", "integer", {">": 0})
```

ValidateOptionTextValue

```
ValidateOptionTextValue(OptionName, OptionValue, ValidValues)
```

Validate option value based on the valid specified values separated by spaces.

Arguments:

```
OptionName (str): Command line option name.
OptionValue (str): Command line option value.
ValidValues (str): Space delimited string containing valid values.
```

The function exits with an error message for an invalid option value.

ValidateOptionsDistinctFileNames

```
ValidateOptionsDistinctFileNames(OptionName1, FilePath1, OptionName2, FilePath2)
```

Validate two distinct file names.

Arguments:

```
OptionName1 (str): Command line option name.
FilePath1 (str): Name of a file with complete file path.
OptionName2 (str): Command line option name.
FilePath2 (str): Name of a file with complete file path.
```

The function exits with an error message for two non distinct file names.

ValidateOptionsOutputFileOverwrite

```
ValidateOptionsOutputFileOverwrite(OptionName, FilePath, OverwriteOptionName,
OverwriteStatus)
```

Validate overwriting of output file.

Arguments:

```
OptionName (str): Command line option name.
FilePath (str): Name of a file with complete file path.
OverwriteOptionName (str): Overwrite command line option name.
OverwriteStatus (bool): True, overwrite
```

The function exits with an error message for a file that is present and is not allowed to be written as indicated by value of OverwriteStatus.

WrapText

```
WrapText(Text, Delimiter, Width)
```

Wrap text using specified delimiter and width.

Arguments:

```
Text (string): Input text
Delimiter (string): Delimiter for wrapping text
Width (int): Max number of characters before wrapping text
```

Returns:

```
str : Wrapped text
```

AUTHOR

Manish Sud <msud@san.rr.com>

COPYRIGHT