NAME

PyMOLUtil

SYNOPSIS

import PyMOLUtil

DESCRIPTION

PyMOLUtil module provides the following functions:

AreAminoAcidResiduesPresent, AreNucleicAcidResiduesPresent, CalculateCenterOfMass, ConvertFileFormat, ConvertPMLFileToPSEFile, GetAminoAcidResiduesInfo, GetChains, GetChainsAndLigandsInfo, GetInorganicResiduesInfo, GetInterfaceChainsResiduesByCAlphaAtomsDistance, GetInterfaceChainsResiduesBySASAChange, GetLargestLigand, GetLigandResiduesInfo, GetLigands, GetMolecules, GetNucleicAcidResiduesInfo, GetPhiPsiCategoriesResiduesInfo, GetPhiPsiChainsAndResiduesInfo, GetPhiPsiResiduesInfo, GetPocketInorganicResiduesInfo, GetPocketPolymerResiduesInfo, GetPocketSolventResiduesInfo, GetPolymerResiduesInfo, GetSelectionResiduesInfo, GetSolventResiduesInfo, GetSurfaceAndBuriedResiduesInfo, GetnterfaceChainsResiduesByHeavyAtomsDistance, ProcessChainSelectionsOptionsInfo, ProcessChainsAndLigandsOptionsInfo, ProcessResidueTypesOptionsInfo, ProcessSurfaceAtomTypesColorsOptionsInfo, SetupPMLForAlignment, SetupPMLForBFactorPuttyView, SetupPMLForBallAndStickView, SetupPMLForEnableDisable, SetupPMLForGroup, SetupPMLForHydrophobicAndChargeSurfaceView, SetupPMLForHydrophobicContactsView, SetupPMLForHydrophobicSurfaceView, SetupPMLForInorganicView, SetupPMLForLigandPocketInorganicView, SetupPMLForLigandPocketSolventView, SetupPMLForLigandPocketView, SetupPMLForLigandView, SetupPMLForPolarContactsView, SetupPMLForPolymerChainComplexView, SetupPMLForPolymerChainView, SetupPMLForPolymerComplexView, SetupPMLForSelectionDisplayView, SetupPMLForSolventView, SetupPMLForSurfaceView, SetupPMLHeaderInfo

FUNCTIONS

AreAminoAcidResiduesPresent

```
AreAminoAcidResiduesPresent(MoleculeName, ChainName, Type = "Any")
```

Check for the presence of amino acid residues in a chain of a molecule. Chains are identified using PyMOL 'polymer' selection. Nonstandard amino acid residues correspond to all residues other than the standard amino acids and nucleic acids. Any amino acid residues cover all residues other than the standard nucleic acids.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
Type (str): Types of amino acids: Standard, NonStandard, Any
```

Returns:

```
boolean: True or False.
```

AreNucleicAcidResiduesPresent

```
AreNucleicAcidResiduesPresent(MoleculeName, ChainName)
```

Check for the presence of nucleic acid residues in a chain of a molecule. Chains are identified using PyMOL 'polymer' selection.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
```

Returns:

```
boolean: True or False.
```

CalculateCenterOfMass

```
CalculateCenterOfMass(Selection = "all", Quiet = 0)
```

Calculate center of mass for a selection.

Arguments:

```
Selection (str): A PyMOL selection.
Quiet (int): Print information.
```

Returns:

```
list: X, Y, Z coordinates for center of mass.
```

ConvertFileFormat

```
ConvertFileFormat(Infile, Outfile, Reinitialize = True, OutputFeedback = True)
```

Convert infile to outfile by automatically detecting their formats from the file extensions.

The formats of both input and output files must be a valid format supported by PyMOL.

Arguments:

```
Infile (str): Name of input file.
Outfile (str): Name of outfile file.
Reinitialize (bool): Reinitialize PyMOL before loading input file.
OutputFeedback (bool): Control output feedback.
```

ConvertPMLFileToPSEFile

```
ConvertPMLFileToPSEFile(PMLFile, PSEFile, Reinitialize = True, OutputFeedback = True)
```

Convert PML file to PME file.

Arguments:

```
PMLFile (str): Name of PML file.
PSEFile (str): Name of PSE file.
Reinitialize (bool): Reinitialize PyMOL before loading PML file.
OutputFeedback (bool): Control output feedback.
```

GetAminoAcidResiduesInfo

```
GetAminoAcidResiduesInfo(MoleculeName, ChainName, Type = "Standard")
```

Get information for amino acid residues present in a chain of a molecule. Chains are identified using PyMOL 'polymer' selection. Nonstandard amino acid residues correspond to all residues other than the standard amino acids and nucleic acids. Any amino acid residues cover all residues other than the standard nucleic acids.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
Type (str): Types of amino acids: Standard, NonStandard, Any
```

Returns:

```
dict: A dictionary containing list of residue names and dictionaries of
    residue numbers and residue count for each residue. Names of
    residues in the dictionary are not sorted.
```

Example(s):

```
ResiduesInfo = GetPolymerResiduesInfo(MolName, ChainName)
for ResName in ResiduesInfo["ResNames"]:
    ResCount = ResiduesInfo["ResCount"][ResName]
    ResNums = ResiduesInfo["ResNum"][ResName]
    MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
        (ResName, ResCount, ResNums))
```

GetChains

```
GetChains(MoleculeName, RemoveEmpty = True)
```

Get chain identifiers present in a molecule.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
RemoveEmpty (bool): Remove empty chain ID from the list of chain IDs
    returned by PyMOL.
```

Returns:

```
list: Names of chains present in a molecule, sorted alphabetically in a
    ascending order.
```

GetChainsAndLigandsInfo

```
GetChainsAndLigandsInfo(Infile, MolName, Quite = False, LigandSortBy = "Size",
LigandSortOrder = "Auto", LigandIgnoreHydrogens = "Yes")
```

Get chain identifiers present in a molecule along with names of the ligands present in chains. Ligands are identified using PyMOL 'organic' selection.

Arguments:

```
Infile (str) : Name of a file.
MolName (str) : Name to use for PyMOL molecule object.
Quite (bool) : Flag
LigandSortBy (str): Sort ligand names alphabetically or by size. Possible
    values: Alphabetical or Size
LigandSortOrder (str): Sort order for sorting ligands. Possible values:
    Ascending, Descending, Auto. The 'Auto' value implies automatic
    determination of sort order based on the value of 'SortBy'.
    Automatic defaults: Descending for SortBy value of Size; Ascending
    for SortBy value of Alphabetical.
LigandIgnoreHydrogens (str): Ignore hydrogens during determination of ligand
    size.
```

Returns:

```
dict: A dictionary containing list of chain identifiers and dictionaries
    of chains containing lists of ligand names for each chain. Names of
    ligands present in chain for a molecule sorted by size or
    alphabetically.
```

Example(s):

```
ChainsAndLigandsInfo = GetChainsAndLigandsInfo(Infile, MolName)
for ChainID in ChainsAndLigandsInfo["ChainIDs"]:
    for LigandID in ChainsAndLigandsInfo["LigandIDs"][ChainID]:
        MiscUtil.PrintInfo("ChainID: %s; LigandID: %s" % (ChainID,
            LigandID))
```

GetInorganicResiduesInfo

```
GetInorganicResiduesInfo(MoleculeName, ChainName)
```

Get information for inorganic residues present in a chain of a molecule. Inorganic residues are identified using PyMOL 'inorganic' selection.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
```

Returns:

```
dict: A dictionary containing list of residue names and dictionaries of
    residue numbers and residue count for each residue. Names of
    residues in the dictionary are not sorted.
```

Example(s):

```
    ResiduesInfo = GetInorganicResiduesInfo(MolName, ChainName)
    for ResName in ResiduesInfo["ResNames"]:
        ResCount = ResiduesInfo["ResCount"][ResName]
        ResNums = ResiduesInfo["ResNum"][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
            (ResName, ResCount, ResNums))
```

GetInterfaceChainsResiduesByCAlphaAtomsDistance

```
    GetInterfaceChainsResiduesByCAlphaAtomsDistance(MoleculeName1, ChainNames1,
MoleculeName2, ChainNames2, DistanceCutoff = 8.0)
```

Get information for interface residues between chains in two molecules based on the distance between CAlpha atoms. The chain specification for molecules may contain multiple chain names delimited by commas.

The interface residues are identified using PyMOL 'bycalpha' selection operator with in a specified distance.

Arguments:

```
    MoleculeName1 (str): Name of a PyMOL molecule object.
    ChainNames1 (str): A chain name or comma delimited list of chain
        names in a molecule.
    MoleculeName2 (str): Name of a PyMOL molecule object.
    ChainNames2 (str): A chain name or comma delimited list of chain
        names in a molecule.
    DistanceCutoff (float): Distance cutoff for distance between
        any two CAlpha atoms in interface residues in different chains.
```

Returns:

```
    dict1: Interface residues in a chain for first molecule. It is a
        dictionary containing list of residue names and dictionaries of
        residue numbers and residue count for each residue. Names of
        residues in the dictionary are not sorted.
    dict2: Interface residues in the chain for second molecule.
```

Example(s):

```
    ResiduesInfo1, ResiduesInfo2 =
        GetInterfaceResiduesByHeavyAtomsDistance(MolName1,
        ChainName1, MolName2, ChainName2, DistanceCutoff)
    for ChainID in ResiduesInfo1["ChainIDs"]:
        for ResName in ResiduesInfo1["ResNames"][ChainID]:
            ResCount = ResiduesInfo1["ResCount"][ChainID][ResName]
            ResNums = ResiduesInfo1["ResNum"][ChainID][ResName]
            MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums:
            %s" % (ResName, ResCount, ResNums))
```

GetInterfaceChainsResiduesBySASAChange

```
    GetInterfaceChainsResiduesBySASAChange(MoleculeName1, ChainNames1, MoleculeName2,
ChainNames2, ChangeCutoff = 0.75)
```

Get information for interface residues between chains in two molecules based on the change in solvent accessible surface area (SASA) of a residue in chains in a molecule and chain complex containing specified chains across both molecules. The chain specification for molecules may contain multiple chain names delimited by commas.

Arguments:

```
    MoleculeName1 (str): Name of a PyMOL molecule object.
    ChainNames1 (str): A chain name or comma delimited list of chain
        names in a molecule.
    MoleculeName2 (str): Name of a PyMOL molecule object.
    ChainNames2 (str): A chain name or comma delimited list of chain
        names in a molecule.
    ChangeCutoff (float): SASA change cutoff for heavy atoms in
```

```
        a interface residue between an individual chain and a complex
        containing both chains.  Units: Angstroms ** 2
```

Returns:

```
    dict1: Interface residues in the chain for first molecule. It is a
        dictionary containing list of residue names and dictionaries of
        residue numbers and residue count for each residue. Names of
        residues in the dictionary are not sorted.
    dict2: Interface residues in the chain for second molecule.
```

Example(s):

```
    ResiduesInfo1, ResiduesInfo2 =
        GetInterfaceResiduesBySASAChange(MolName1,
        ChainName1, MolName2, ChainName2, DistanceCutoff)
    for ResName in ResiduesInfo1["ResNames"]:
        ResCount = ResiduesInfo1["ResCount"][ResName]
        ResNums = ResiduesInfo1["ResNum"][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
            (ResName, ResCount, ResNums))
```

GetLargestLigand

```
    GetLargestLigand(MoleculeName, ChainName, IgnoreHydrogens = 'Yes')
```

Get name of the largest ligand for a chain present in a molecule. Ligands are identified using PyMOL 'organic' selection.

Arguments:

```
    IgnoreHydrogens (str): Ignore hydrogens during determination of ligand
        size.
```

Returns:

```
    str: Name of the largest ligand present in a chain.
```

GetLigandResiduesInfo

```
    GetLigandResiduesInfo(MoleculeName, ChainName)
```

Get information for ligand residues present in a chain of a molecule. Ligands are identified using PyMOL 'organic' selection.

Arguments:

```
    MoleculeName (str): Name of a PyMOL molecule object.
    ChainName (str): Name of a chain in a molecule.
```

Returns:

```
    dict: A dictionary containing list of residue names and dictionaries of
        residue numbers and residue count for each residue. Names of
        residues in the dictionary are not sorted.
```

Example(s):

```
    ResiduesInfo = GetLigandResiduesInfo(MolName, ChainName)
    for ResName in ResiduesInfo["ResNames"]:
        ResCount = ResiduesInfo["ResCount"][ResName]
        ResNums = ResiduesInfo["ResNum"][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
            (ResName, ResCount, ResNums))
```

GetLigands

```
    GetLigands(MoleculeName, ChainName, SortBy = "Size", SortOrder = "Auto",
IgnoreHydrogens = "Yes")
```

Get names of ligands present in a chain of a molecule. Ligands are identified using PyMOL 'organic' selection.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
SortBy (str): Sort ligand names alphabetically or by size. Possible
    values: Alphabetical or Size
SortOrder (str): Sort order for sorting ligands. Possible values:
    Ascending, Descending, Auto. The 'Auto' value implies automatic
    determination of sort order based on the value of 'SortBy'.
    Automatic defaults: Descending for SortBy value of Size; Ascending
    for SortBy value of Alphabetical.
IgnoreHydrogens (str): Ignore hydrogens during determination of ligand
    size.
```

Returns:

```
list: Names of ligands present in chain for a molecule sorted by size
    or alphabetically.
```

GetMolecules

```
GetMolecules(Selection = "all")
```

Get names of molecule objects in a selection or all molecule objects.

Arguments:

```
Selection: (str): A PyMOL selection.
```

Returns:

```
list: Names of molecule objects.
```

GetNucleicAcidResiduesInfo

```
GetNucleicAcidResiduesInfo(MoleculeName, ChainName)
```

Get information for nucleic acid residues present in a chain of a molecule. Chains are identified using PyMOL 'polymer' selection.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
```

Returns:

```
dict: A dictionary containing list of residue names and dictionaries of
    residue numbers and residue count for each residue. Names of
    residues in the dictionary are not sorted.
```

Example(s):

```
ResiduesInfo = GetPolymerResiduesInfo(MolName, ChainName)
for ResName in ResiduesInfo["ResNames"]:
    ResCount = ResiduesInfo["ResCount"][ResName]
    ResNums = ResiduesInfo["ResNum"][ResName]
    MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
        (ResName, ResCount, ResNums))
```

GetPhiPsiCategoriesResiduesInfo

```
GetPhiPsiCategoriesResiduesInfo(MoleculeName, ChainName)
```

Get phi and psi torsion angle information for residues in a chain of a molecule containing amino acids.

The phi and psi angles are optionally categorized into the following groups corresponding to four types of Ramachandran plots:

General: All residues except glycine, proline, or pre-proline Glycine: Only glycine residues Proline: Only proline residues Pre-Proline: Only residues before proline not including glycine or proline

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
```

Returns:

```
dict1: Phi and psi angle information for residues in General category.
    It's a dictionary containing sorted list of residue numbers and
    dictionaries of residue names, phi and psi angles for each residue
    number.
dict2: Phi and psi angle information for residues in Gly category.
dict3: Phi and psi angle information for residues in Pro category.
dict2: Phi and psi angle information for residues in Pre-Pro category.
```

Example(s):

```
GeneralPhiPsiInfo, GlyPhiPsiInfo, ProPhiPsiInfo, PreProPhiPsiInfo =
    GetPhiPsiCategoriesResiduesInfo(MolName, ChainID)
for ResNum in GeneralPhiPsiInfo["ResNums"]:
    ResName = GeneralPhiPsiInfo["ResName"][ResNum]
    Phi = GeneralPhiPsiInfo["Phi"][ResNum]
    Psi = GeneralPhiPsiInfo["Psi"][ResNum]
    MiscUtil.PrintInfo("ResNum: %s; ResName: %s;
        Phi: %8.2f; Psi: %8.2f" % (ResNum, ResName, Phi, Psi))
```

GetPhiPsiChainsAndResiduesInfo

```
GetPhiPsiChainsAndResiduesInfo(MoleculeName, Categorize = True)
```

Get phi and psi torsion angle information for residues across chains in a molecule containing amino acids.

The phi and psi angles are optionally categorized into the following groups corresponding to four types of Ramachandran plots:

General: All residues except glycine, proline, or pre-proline Glycine: Only glycine residues Proline: Only proline residues Pre-Proline: Only residues before proline not including glycine or proline

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
```

Returns:

```
dict: A dictionary containing sorted list of residue numbers for each
    chain and dictionaries of residue names, phi and psi angles for each
    residue number.
```

Example(s):

```
PhiPsiInfoMap = GetPhiPsiChainsAndResiduesInfo(MolName)
for ChainID in PhiPsiInfoMap["ChainIDs"]:
    for ResNum in PhiPsiInfoMap["ResNums"][ChainID]:
        ResName = PhiPsiInfoMap["ResName"][ChainID][ResNum]
        Phi = PhiPsiInfoMap["Phi"][ChainID][ResNum]
        Psi = PhiPsiInfoMap["Psi"][ChainID][ResNum]
        Category = PhiPsiInfoMap["Category"][ChainID][ResNum]
        MiscUtil.PrintInfo("ChainID: %s; ResNum: %s; ResName: %s; Phi: %8.2f;
            Psi: %8.2f; Category: %s" % (ChainID, ResNum, ResName, Phi,
            Psi, Category))
```

GetPhiPsiResiduesInfo

```
GetPhiPsiResiduesInfo(MoleculeName, ChainName, Categorize = True)
```

Get phi and psi torsion angle information for residues in a chain of a molecule containing amino acids.

The phi and psi angles are optionally categorized into the following groups corresponding to four types of

Ramachandran plots:

General: All residues except glycine, proline, or pre-proline Glycine: Only glycine residues Proline: Only proline residues Pre-Proline: Only residues before proline not including glycine or proline

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
```

Returns:

```
dict: A dictionary containing sorted list of residue numbers and
    dictionaries of residue names, phi and psi angles for each residue
    number.
```

Example(s):

```
PhiPsiInfoMap = GetPhiPsiResiduesInfo(MolName, ChainName, True)
for ResNum in PhiPsiInfoMap["ResNums"]:
    ResName = PhiPsiInfoMap["ResName"][ResNum]
    Phi = PhiPsiInfoMap["Phi"][ResNum]
    Psi = PhiPsiInfoMap["Psi"][ResNum]
    Category = PhiPsiInfoMap["Category"][ResNum]
    MiscUtil.PrintInfo("ResNum: %s; ResName: %s; Phi: %8.2f;
        Psi: %8.2f; Category: %s" % (ResNum, ResName, Phi, Psi,
        Categorize))
```

GetPocketInorganicResiduesInfo

```
GetPocketInorganicResiduesInfo(MoleculeName, ChainName, LigandResName, LigandResNum,
PocketDistanceCutoff)
```

Get information for inorganic residues present in a pocket around a ligand in a molecule. Inorganic residues are identified using PyMOL 'inorganic' selection.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
LigandResName (str): Residue name of a ligand in a chain.
LigandResNum (str): Residue number of a ligand in a chain.
PocketDistanceCutoff (float): Distance around a ligand to identify
    pocket residues.
```

Returns:

```
dict: A dictionary containing list of residue names and dictionaries of
    residue numbers and residue count for each residue. Names of
    residues in the dictionary are not sorted.
```

Example(s):

```
ResiduesInfo = GetPocketInorganicResiduesInfo(MolName, ChainName)
for ResName in ResiduesInfo["ResNames"]:
    ResCount = ResiduesInfo["ResCount"][ResName]
    ResNums = ResiduesInfo["ResNum"][ResName]
    MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
        (ResName, ResCount, ResNums))
```

GetPocketPolymerResiduesInfo

```
GetPocketPolymerResiduesInfo(MoleculeName, ChainName, LigandResName, LigandResNum,
PocketDistanceCutoff)
```

Get information for chain residues present in a pocket around a ligand in a molecule. Polymer residues are identified using negation of PyMOL selection operators 'organic', 'solvent', and 'inorganic'.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
```

```
    ChainName (str): Name of a chain in a molecule.
    LigandResName (str): Residue name of a ligand in a chain.
    LigandResNum (str): Residue number of a ligand in a chain.
    PocketDistanceCutoff (float): Distance around ligand to identify pocket
        residues.
```

Returns:

```
    dict: A dictionary containing list of residue names and dictionaries of
        residue numbers and residue count for each residue. Names of
        residues in the dictionary are not sorted.
```

Example(s):

```
    ResiduesInfo = GetPocketPolymerResiduesInfo(MolName, ChainName)
    for ResName in ResiduesInfo["ResNames"]:
        ResCount = ResiduesInfo["ResCount"][ResName]
        ResNums = ResiduesInfo["ResNum"][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
            (ResName, ResCount, ResNums))
```

GetPocketSolventResiduesInfo

```
    GetPocketSolventResiduesInfo(MoleculeName, ChainName, LigandResName, LigandResNum,
PocketDistanceCutoff)
```

Get information for solvent residues present in a pocket around a ligand in a molecule. Solvent residues are identified using PyMOL 'solvent' selection.

Arguments:

```
    MoleculeName (str): Name of a PyMOL molecule object.
    ChainName (str): Name of a chain in a molecule.
    LigandResName (str): Residue name of a ligand in a chain.
    LigandResNum (str): Residue number of a ligand in a chain.
    PocketDistanceCutoff (float): Distance around ligand to identify pocket
        residues.
```

Returns:

```
    dict: A dictionary containing list of residue names and dictionaries of
        residue numbers and residue count for each residue. Names of
        residues in the dictionary are not sorted.
```

Example(s):

```
    ResiduesInfo = GetPocketSolventResiduesInfo(MolName, ChainName)
    for ResName in ResiduesInfo["ResNames"]:
        ResCount = ResiduesInfo["ResCount"][ResName]
        ResNums = ResiduesInfo["ResNum"][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
            (ResName, ResCount, ResNums))
```

GetPolymerResiduesInfo

```
    GetPolymerResiduesInfo(MoleculeName, ChainName)
```

Get information for residues present in a chain of a molecule. Chains are identified using PyMOL 'polymer' selection.

Arguments:

```
    MoleculeName (str): Name of a PyMOL molecule object.
    ChainName (str): Name of a chain in a molecule.
```

Returns:

```
    dict: A dictionary containing list of residue names and dictionaries of
        residue numbers and residue count for each residue. Names of
```

```
        residues in the dictionary are not sorted.
```

Example(s):

```
    ResiduesInfo = GetPolymerResiduesInfo(MolName, ChainName)
    for ResName in ResiduesInfo["ResNames"]:
        ResCount = ResiduesInfo["ResCount"][ResName]
        ResNums = ResiduesInfo["ResNum"][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
            (ResName, ResCount, ResNums))
```

GetSelectionResiduesInfo

```
    GetSelectionResiduesInfo(SelectionCmd)
```

Get information for residues in a chain specified by a selection command.

Arguments:

```
    SelectionCmd (str): PyMOL selection command.
```

Returns:

```
    dict: A dictionary containing list of residue names and dictionaries of
        residue numbers and residue count for each residue. Names of
        residues in the dictionary are not sorted.
```

Example(s):

```
    ResiduesInfo = GetSelectionResiduesInfo(SelectionCmd)
    for ResName in ResiduesInfo["ResNames"]:
        ResCount = ResiduesInfo["ResCount"][ResName]
        ResNums = ResiduesInfo["ResNum"][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
            (ResName, ResCount, ResNums))
```

GetSolventResiduesInfo

```
    GetSolventResiduesInfo(MoleculeName, ChainName)
```

Get information for solvent residues present in a chain of a molecule. Solvents are identified using PyMOL 'solvent' selection.

Arguments:

```
    MoleculeName (str): Name of a PyMOL molecule object.
    ChainName (str): Name of a chain in a molecule.
```

Returns:

```
    dict: A dictionary containing list of residue names and dictionaries of
        residue numbers and residue count for each residue. Names of
        residues in the dictionary are not sorted.
```

Example(s):

```
    ResiduesInfo = GetSolventResiduesInfo(MolName, ChainName)
    for ResName in ResiduesInfo["ResNames"]:
        ResCount = ResiduesInfo["ResCount"][ResName]
        ResNums = ResiduesInfo["ResNum"][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
            (ResName, ResCount, ResNums))
```

GetSurfaceAndBuriedResiduesInfo

```
    GetSurfaceAndBuriedResiduesInfo(MoleculeName, ChainName, SASACutoff = 2.5)
```

Get information for surafce and buried residues present in a chain of a molecule. The surface residues correspond to residues with Solvent Accessible Surface Area (SASA) greater than or equal to the cutoff value. Otherwise, these residues are considered as buried residues.

Arguments:

```
MoleculeName (str): Name of a PyMOL molecule object.
ChainName (str): Name of a chain in a molecule.
SASACutoff (float): SASA cutoff for heavy atoms corresponding to
    surface residues in chain. Units: Angstroms ** 2
```

Returns:

```
dict: A dictionary containing list of residue names and dictionaries of
    residue numbers and residue count for each residue. Names of
    residues in the dictionary are not sorted.
dict2: Buried residues in a chain.
```

Example(s):

```
SurfaceResiduesInfo, BurriedResiduesInfo =
    GetSurfaceResiduesInfo(MolName, ChainName, 2.5)
for ResName in SurfaceResiduesInfo["ResNames"]:
    ResCount = ResiduesInfo["ResCount"][ResName]
    ResNums = ResiduesInfo["ResNum"][ResName]
    MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums: %s" %
        (ResName, ResCount, ResNums))
```

GetnterfaceChainsResiduesByHeavyAtomsDistance

```
GetnterfaceChainsResiduesByHeavyAtomsDistance(MoleculeName1, ChainNames1,
MoleculeName2, ChainNames2, DistanceCutoff = 5.0)
```

Get information for interface residues between chains in two molecules based on the distance between heavy atoms. The chain specification for molecules may contain multiple chain names delimited by commas.

The interface residues are identified using PyMOL 'byresidue' selection operator with in a specified distance.

Arguments:

```
MoleculeName1 (str): Name of a PyMOL molecule object.
ChainNames1 (str): A chain name or comma delimited list of chain
    names in a molecule.
MoleculeName2 (str): Name of a PyMOL molecule object.
ChainNames2 (str): A chain name or comma delimited list of chain
    names in a molecule.
DistanceCutoff (float): Distance cutoff for distance between
    any two heavy atoms in interface residues in different chains.
```

Returns:

```
dict1: Interface residues in a chain for first molecule. It is a
    dictionary containing list of residue names and dictionaries of
    residue numbers and residue count for each residue. Names of
    residues in the dictionary are not sorted.
dict2: Interface residues in the chain for second molecule.
```

Example(s):

```
ResiduesInfo1, ResiduesInfo2 =
    GetInterfaceResiduesByHeavyAtomsDistance(MolName1,
    ChainName1, MolName2, ChainName2, DistanceCutoff)
for ChainID in ResiduesInfo1["ChainIDs"]:
    for ResName in ResiduesInfo1["ResNames"][ChainID]:
        ResCount = ResiduesInfo1["ResCount"][ChainID][ResName]
        ResNums = ResiduesInfo1["ResNum"][ChainID][ResName]
        MiscUtil.PrintInfo("ResName: %s; ResCount: %s; ResNums:
        %s" % (ResName, ResCount, ResNums))
```

ProcessChainSelectionsOptionsInfo

```
ProcessChainSelectionsOptionsInfo(SelectionOptionName, SelectionOptionValue)
```

Process names and selections specified using command line option. It is a pairwise list comma delimited values corresponding to PyMOL object names and selection specification

Arguments:

```
SelectionOptionName (str): Name of command line  option.
SelectionOptionValue (str): Value for command line option.
```

Returns:

```
dict: A dictionary containing lists f names and selection commands.
```

Example(s):

```
ChainSelectionsInfo =
    PyMOLUtil.ProcessChainSelectionsOptionsInfo("--selectionsChain",
    OptionsInfo["SelectionChains"])
```

ProcessChainsAndLigandsOptionsInfo

```
ProcessChainsAndLigandsOptionsInfo(ChainsAndLigandsInfo, ChainsOptionName,
ChainsOptionValue, LigandsOptionName = None, LigandsOptionValue = None)
```

Process specified chain and ligand IDs using command line options.

Arguments:

```
ChainsAndLigandsInfo (dict): A dictionary containing information
    existing chains and ligands.
ChainsOptionName (str): Name of command line chains option.
ChainsOptionValue (str): Value for command line chains option.
LigandsOptionName (str): Name of command line ligands option.
LigandsOptionValue (str): Value for command line ligands option.
```

Returns:

```
dict: A dictionary containing list of chain identifiers and dictionaries
    of chains containing lists of ligand names for each chain.
```

Example(s):

```
ChainsAndLigandsInfo = ProcessChainsAndLigandsOptionsInfo(
    ChainsAndLigandsInfo, "-c, --chainIDs", OptionsInfo["ChainIDs"],
    "-l, --ligandIDs", OptionsInfo["LigandIDs"])
for ChainID in ChainsAndLigandsInfo["ChainIDs"]:
    for LigandID in ChainsAndLigandsInfo["LigandIDs"][ChainID]:
        MiscUtil.PrintInfo("ChainID: %s; LigandID: %s" % (ChainID,
            LigandID))
```

ProcessResidueTypesOptionsInfo

```
ProcessResidueTypesOptionsInfo(ResidueTypesOptionName, ResidueTypesOptionValue)
```

Process specified residue types using command line option.

Arguments:

```
ResidueTypesOptionName (str): Name of command line  option.
ResidueTypesOptionValue (str): Value for command line option.
```

Returns:

```
list: A list containing names of valid residue types.
dict: A dictionary containing residue types pointing to dictionaries of
    color names and list of residues for a residue type.
```

Example(s):

```
ResidueTypesNamesInfo, ResidueTypesParamsInfo =
    ProcessChainsAndLigandsOptionsInfo("-r, --residueTypes",
    OptionsInfo["ResidueTypes"])
```

```
    for ResidueTypeName in ResidueTypesNamesInfo:
        MiscUtil.PrintInfo("ResidueType: %s; Color: %s; Residues: %s" %
        (ResidueTypeName, ResidueTypeName[ResidueTypeName]["Color"],
        " ".join(ResidueTypeName[ResidueTypeName]["Residues"]))
```

ProcessSurfaceAtomTypesColorsOptionsInfo

```
    ProcessSurfaceAtomTypesColorsOptionsInfo(ColorOptionName, ColorOptionValue)
```

Process specified surafce atom types colors using command line option.

Arguments:

```
    ColorOptionName (str): Name of command line  option.
    ColorOptionValue (str): Value for command line option.
```

Returns:

```
    dict: A dictionary containing atom types and colors.
```

Example(s):

```
    AtomTypesColorNamesInfo =
        PyMOLUtil.ProcessSurfaceAtomTypesColorsOptionsInfo(
        "--surfaceAtomTypesColors", OptionsInfo["SurfaceAtomTypesColors"])
```

SetupPMLForAlignment

```
    SetupPMLForAlignment(Method, RefSelection, FitSelection)
```

Setup PML commands for aligning a pair of selection using a specified alignment method.

Arguments:

```
    Method (str): Alignment method. Possible values: align, cealign, super.
    RefSelection (str): Name of reference selection which stays stationary.
    FitSelection (str): Name of selection to align to reference selection.
```

Returns:

```
    str: PML commands for aligning  a pair of selections.
```

SetupPMLForBFactorPuttyView

```
    SetupPMLForBFactorPuttyView(Name, Selection, ColorPalette = "blue_white_red", Enable =
True)
```

Setup PML commands for creating a B factor putty view for a specified selection. The B factor values must be available for the atoms. The atoms are colored using a color spectrum corresponding to a specified color palette. Any valid PyMOL color palette name may be used.

Arguments:

```
    Name (str): Name of a new PyMOL B factor putty object.
    Selection (str): Name of PyMOL selection.
    ColorPalette (str): Name of color palette to use for color spectrum.
    Enable (bool): Display status of B factor putty object.
```

Returns:

```
    str: PML commands for B factor putty view.
```

SetupPMLForBallAndStickView

```
    SetupPMLForBallAndStickView(Name, Selection, Enable = True, SphereScale = 0.3,
StickRadius = 0.2)
```

Setup PML commands for creating a ball and stick view for a specified selection.

Arguments:

```
    Name (str): Name of a new PyMOL ball and stick object.
```

```
    Selection (str): Name of PyMOL selection.
    Enable (bool): Display status of ball and stick object.
    SphereScale (float): Scaling factor for sphere radii.
    StickScale (float): Scaling factor for stick radii.
```

Returns:

```
    str: PML commands for ball and stick view.
```

SetupPMLForEnableDisable

```
    SetupPMLForEnableDisable(Name, Enable = True)
```

Setup PML command for enabling or disabling display of a PyMOL object.

Arguments:

```
    Name (str): Name of a PyMOL object.
    Enable (bool): Display status.
```

Returns:

```
    str: PML command for enabling or disabling display of an object.
```

SetupPMLForGroup

```
    SetupPMLForGroup(GroupName, GroupMembersList, Enable = None, Action = None)
```

Setup PML commands for creating a group from a list of group members. The display and open status of the group may be optionally set. The 'None' values for Enable and Action imply usage of PyMOL defaults for the creation of group.

Arguments:

```
    GroupName (str): Name of a PyMOL group.
    GroupMembersList (list): List of group member names.
    Enable (bool): Display status of group.
    Action (str): Open or close status of group object.
```

Returns:

```
    str: PML commands for creating a group object.
```

SetupPMLForHydrophobicAndChargeSurfaceView

```
    SetupPMLForHydrophobicAndChargeSurfaceView(Name, Selection, HydrophobicAtomsColor =
"yellow", NegativelyChargedAtomsColor = "red", PositivelyChargedAtomsColor = "blue",
OtherAtomsColor = "gray90", Enable = True, DisplayAs = "cartoon")
```

Setup PML commands for creating a surface colored by hydrophobic and charge [REF 140] properties of atoms in amino acids. The atom names in standard amino acid residues are used to identify atom types as shown below:

Hydrophobic: C atoms not bound to N or O atoms; NegativelyCharged: Side chain O atoms in ASP and GLU; PositivelyCharged: Side chain N atoms in ARG and LYS; Others: Remaining atoms in polar and other residues

The amino acid atom names to color specific atoms are taken from YRB.py script [REF 140]. The color values may also be specified as comma delimited RGB triplets. For example: HydrophobicAtomsColor = "0.950 0.78 0.0", NegativelyChargedAtomsColor = "1.0 0.4 0.4", PositivelyChargedAtomsColor = "0.2 0.5 0.8", OtherAtomsColor = "0.95 0.95 0.95"

Arguments:

```
    Name (str): Name of a new PyMOL hydrophobic surface object.
    Selection (str): Name of PyMOL selection.
    HydrophobicAtomsColor (str): Color name or space delimited RGB values
    NegativelyChargedAtomsColor (str): Color name or space delimited RGB values
    PositivelyChargedAtomsColor (str): Color name or space delimited RGB values
    OtherAtomsColor (str): Color name or space delimited RGB values
    Enable (bool): Display status of surface object.
```

```
    DisplayAs (str): Any additional valid display type such as lines,
        sticks, ribbon, cartoon, or None.
```

Returns:

```
    str: PML commands for hydrophobic and charge surface view.
```

SetupPMLForHydrophobicContactsView

```
    SetupPMLForHydrophobicContactsView(Name, Selection1, Selection2, Enable = True, Color =
"yellow", Cutoff = None)
```

Setup PML commands for creating hydrophobic contacts view between a pair of selections. The hydrophobic contacts are shown between pairs of carbon atoms not connected to hydrogen bond donor or acceptors atoms as identified by PyMOL. The distance labels are shown by default.

Arguments:

```
    Name (str): Name of a new PyMOL polar contacts object.
    Selection1 (str): First PyMOL selection.
    Selection2 (str): Second PyMOL selection.
    Enable (bool): Display status of polar contacts object.
    Color (str): Color for polar contact lines and labels.
    Cutoff (float): None or distance cutoff for hydrophobic contacts.
```

Returns:

```
    str: PML commands for polar contacts view between a pair of selections.
```

SetupPMLForHydrophobicSurfaceView

```
    SetupPMLForHydrophobicSurfaceView(Name, Selection, ColorPalette = "RedToWhite", Enable
= True, DisplayAs = "cartoon")
```

Setup PML commands for creating a hydrophobic surface view for a specified selection. The surfaces are colored using a specified color palette. This is only valid for amino acids.

Arguments:

```
    Name (str): Name of a new PyMOL hydrophobic surface object.
    Selection (str): Name of PyMOL selection.
    ColorPalette (str): Name of color palette to use for coloring surfaces.
        Possible values: RedToWhite or WhiteToGreen for most hydrophobic
        to least hydrophobic amino acids.
    Enable (bool): Display status of surface object.
    DisplayAs (str): Any additional valid display type such as lines,
        sticks, ribbon, cartoon, or None.
```

Returns:

```
    str: PML commands for hydrophobic surface view.
```

SetupPMLForInorganicView

```
    SetupPMLForInorganicView(Name, Selection, Enable = True)
```

Setup PML commands for creating a inorganic view corresponding to inorganic residues present in a selection. The inorganic residues are identified using inorganic selection operator available in PyMOL. The inorganic residues are displayed as 'lines' and 'nonbonded'.

Arguments:

```
    Name (str): Name of a new PyMOL inorganic object.
    Selection (str): Name of PyMOL selection.
    Enable (bool): Display status of inorganic object.
```

Returns:

```
    str: PML commands for inorganic view.
```

SetupPMLForLigandPocketInorganicView

```
    SetupPMLForLigandPocketInorganicView(Name, Selection, LigandSelection, DistanceCutoff,
Enable = True)
```

Setup PML commands for creating a ligand binding pocket view corresponding to only inorganic residues present in a selection within a specified distance from a ligand selection. The inorganic pocket residues are shown as 'lines' and 'nonbonded'.

Arguments:

```
    Name (str): Name of a new PyMOL solvent binding pocket object.
    Selection (str): PyMOL selection containing binding pocket residues.
    LigandSelection (str): PyMOL selection containing ligand.
    DistanceCutoff (float): Distance cutoff from ligand for selecting
        binding pocket inorganic residues.
    Enable (bool): Display status of binding pocket object.
```

Returns:

```
    str: PML commands for a ligand binding pocket view only showing inorganic
        residues.
```

SetupPMLForLigandPocketSolventView

```
    SetupPMLForLigandPocketSolventView(Name, Selection, LigandSelection, DistanceCutoff,
Enable = True)
```

Setup PML commands for creating a ligand binding pocket view corresponding to only solvent residues present in a selection within a specified distance from a ligand selection. The solvent pocket residues are shown as 'lines' and 'nonbonded'.

Arguments:

```
    Name (str): Name of a new PyMOL solvent binding pocket object.
    Selection (str): PyMOL selection containing binding pocket residues.
    LigandSelection (str): PyMOL selection containing ligand.
    DistanceCutoff (float): Distance cutoff from ligand for selecting
        binding pocket solvent residues.
    Enable (bool): Display status of binding pocket object.
```

Returns:

```
    str: PML commands for a ligand binding pocket view only showing solvent
        residues.
```

SetupPMLForLigandPocketView

```
    SetupPMLForLigandPocketView(Name, Selection, LigandSelection, DistanceCutoff, Enable =
True)
```

Setup PML commands for creating a ligand binding pocket view corresponding all residues present in a selection within a specified distance from a ligand selection. The solvent and inorganic portions of the selection are not included in the binding pocket. The pocket residues are shown as 'lines'. The hydrogen atoms are not displayed.

Arguments:

```
    Name (str): Name of a new PyMOL binding pocket object.
    Selection (str): PyMOL selection containing binding pocket residues.
    LigandSelection (str): PyMOL selection containing ligand.
    DistanceCutoff (float): Distance cutoff from ligand for selecting
        binding pockect residues.
    Enable (bool): Display status of binding pocket object.
```

Returns:

```
    str: PML commands for a ligand binding pocket view.
```

SetupPMLForLigandView

```
SetupPMLForLigandView(Name, Selection, LigandResName, Enable = True)
```

Setup PML commands for creating a ligand view corresponding to a ligand present in a selection. The ligand is identified using organic selection operator available in PyMOL in conjunction with the specified ligand ID. The ligand is colored by atom types and displayed as 'sticks'.

Arguments:

```
Name (str): Name of a new PyMOL ligand object.
Selection (str): PyMOL selection containing ligand.
LigandResName (str): Ligand ID.
Enable (bool): Display status of ligand object.
```

Returns:

```
str: PML commands for a ligand view.
```

SetupPMLForPolarContactsView

```
SetupPMLForPolarContactsView(Name, Selection1, Selection2, Enable = True, Color =
"yellow", Cutoff = None)
```

Setup PML commands for creating polar contacts view between a pair of selections. The polar contact view is generated using 'util.dist' command. The distance labels are shown by default.

Arguments:

```
Name (str): Name of a new PyMOL polar contacts object.
Selection1 (str): First PyMOL selection.
Selection2 (str): Second PyMOL selection.
Enable (bool): Display status of polar contacts object.
Color (str): Color for polar contact lines and labels.
DistanceCutoff (float): None or distance cutoff for polar contacts.
```

Returns:

```
str: PML commands for polar contacts view between a pair of selections.
```

SetupPMLForPolymerChainComplexView

```
SetupPMLForPolymerChainComplexView(ChainComplexName, Selection, ChainName, Enable =
True, ShowSolvent = True, ShowInorganic = True, ShowLines = True)
```

Setup PML commands for creating a polymer chain complex view for a specified chain in a selection. The solvent and inorganic residues are also shown by default. The polymer chain is displayed as 'cartoon'. The 'line' display for the polymer chain is also shown and may be turned off. The organic residues are displayed as 'sticks'. The solvent and inorganic residues are displayed as 'nonbonded' and 'lines'.

Arguments:

```
ChainComplexName (str): Name of a new PyMOL polymer chain complex.
Selection (str): Name of PyMOL selection.
ChainName (str): Name of a chain.
Enable (bool): Display status of chain object.
ShowSolvent (bool): Display solvent residues.
ShowInorganic (bool): Display inorganic residues.
ShowLines (bool): Display lines for polymer chain.
```

Returns:

```
str: PML commands for polymer chain complex view.
```

SetupPMLForPolymerChainView

```
SetupPMLForPolymerChainView(Name, Selection, Enable = True)
```

Setup PML commands for creating a polymer chain view corresponding to backbone and sidechain residues in a selection. The polymer chain is displayed as 'cartoon'.

Arguments:

```
    Name (str): Name of a new PyMOL polymer chain object.
    Selection (str): Name of PyMOL selection.
    Enable (bool): Display status of chain object.
```

Returns:

```
    str: PML commands for polymer chain view.
```

SetupPMLForPolymerComplexView

```
    SetupPMLForPolymerComplexView(MoleculeName, PDBFile, Enable = True, ShowSolvent = True,
ShowInorganic = True, ShowLines = True)
```

Setup PML commands for creating a polymer complex view for all chains in a PDB file. The solvent and inorganic residues are also shown by default. The polymer chains are displayed as 'cartoon'. The 'line' display for the polymer chains is also shown and may be turned off. The organic residues are displayed as 'sticks'. The solvent and inorganic residues are displayed as 'nonbonded' and 'lines'.

Arguments:

```
    MoleculeName (str): Name of a new PyMOL molecule object.
    PDBFile (str): Name of PDB file.
    Enable (bool): Display status of chain object.
    ShowSolvent (bool): Display solvent residues.
    ShowInorganic (bool): Display inorganic residues.
    ShowLines (bool): Display lines for polymer chains.
```

Returns:

```
    str: PML commands for polymer complex view.
```

SetupPMLForSelectionDisplayView

```
    SetupPMLForSelectionDisplayView(Name, Selection, DisplayAs, Color = None, Enable =
True)
```

Setup PML commands for creating a specific molecular display view for a selection.

Arguments:

```
    Name (str): Name of a new PyMOL object.
    Selection (str): Name of PyMOL selection.
    DisplayAs (str): Any valid display type such as lines, sticks, ribbon,
        cartoon, or surface
    Color (str): Color name or use default color.
    Enable (bool): Display status of object.
```

Returns:

```
    str: PML commands for molecular selection view.
```

SetupPMLForSolventView

```
    SetupPMLForSolventView(Name, Selection, Enable = True)
```

Setup PML commands for creating a solvent view corresponding to solvent residues present in a selection. The solvent residues are identified using solvent selection operator available in PyMOL. The solvent residues are displayed as 'nonbonded'.

Arguments:

```
    Name (str): Name of a new PyMOL solvent object.
    Selection (str): Name of PyMOL selection.
    Enable (bool): Display status of inorganic object.
```

Returns:

```
    str: PML commands for solvent view.
```

SetupPMLForSurfaceView

```
    SetupPMLForSurfaceView(Name, Selection, Enable = True, DisplayAs = "cartoon", Color =
"None")
```

Setup PML commands for creating a molecular surface view for a specified selection.

Arguments:

```
    Name (str): Name of a new PyMOL molecular surface object.
    Selection (str): Name of PyMOL selection.
    Enable (bool): Display status of surface object.
    DisplayAs (str): Any additional valid display type such as lines,
        sticks, ribbon, cartoon, or None.
    Color (str): Surafce color.
```

Returns:

```
    str: PML commands for molecular surface view.
```

SetupPMLHeaderInfo

```
    SetupPMLHeaderInfo(ScriptName = None, IncludeLocalPython = True)
```

Setup header information for generating PML files. The local Python functions are optionally embedded in the header information for their use in PML files.

Arguments:

```
    ScriptName (str): Name of script calling the function.
    IncludeLocalPython (bool): Include local Python functions.
```

Returns:

```
    str: Text containing header information for generating PML files.
```

AUTHOR

Manish Sud <msud@san.rr.com>

COPYRIGHT

The functionality available in this file is implemented using PyMOL, a molecular visualization system on an open source foundation originally developed by Warren DeLano.

This file is part of MayaChemTools.