```
#-----------------------------------------------------------------------------
# Copyright (c) Bentley Systems, Incorporated. All rights reserved.
# See COPYRIGHT.md in the repository root for full copyright notice
#-----------------------------------------------------------------------------
from .openStaadHelper import *
from comtypes import automation
from comtypes import CoInitialize
from .osgeometry import OSGeometry
from pathlib import Path
```

[docs]

```
class OSView:
    CoInitialize()
```

[docs]

```
    def __init__(self, staadObj):
        self._staad = staadObj
        self._view = self._staad.View
        self._geometry = OSGeometry(staadObj)

        self._functions= [
            "RefreshView",
            "ShowAllMembers",
            "HideAllMembers",
            "ZoomExtentsMainView",
            "ShowMembers",
            "HideMember",
            "HideMembers",
            "ShowBack",
            "ShowBottom",
            "ShowFront",
            "ShowIsometric",
            "ShowLeft",
            "ShowPlan",
            "ShowRight",
            "SpinLeft",
            "SpinRight",
            "ZoomAll",
            "GetApplicationDesktopSize",
            "SetWindowPosition",
            "RotateUp",
            "RotateDown",
            "RotateLeft",
            "RotateRight",
            "CreateNewViewForSelections",
            "SetLabel",
            "SetSectionView",
            "SetDiagramMode",
            "SetNodeAnnotationMode",
            "SetReactionAnnotationMode",
            "GetInterfaceMode",
            "SetInterfaceMode",
            "SetModeSectionPage",
            "SetBeamAnnotationMode",
            "ShowMember",
            "SetUnits",
            "HidePlate",
            "HideSolid",
            "HideSurface",
            "HideEntity",
            "SelectMembersParallelTo",
            "SelectGroup",
            "SelectInverse",
            "SelectByItemList",
            "SelectByMissingAttribute",
            "SelectEntitiesConnectedToNode",
            "SelectEntitiesConnectedToMember",
            "SelectEntitiesConnectedToPlate",
            "SelectEntitiesConnectedToSolid",
            "GetNoOfBeamsInView",
            "GetBeamsInView",
            "CreateNewViewForSelectionsEx",
            "ExportView",
            "CopyPicture",
            "GetScaleValues",
            "SetScaleValues",
            "GetScaleValueByType",
            "SetScaleValueByType",
            "GetScaleCount",
            "DetachView",
            "RenameView",
            "OpenView",
            "SaveView",
            "GetWindowTitle",
            "GetWindowCount",
            "CloseActiveWindow",
            "SetActiveWindow",
            "SetDesignResults"
        ]

        for function_name in self._functions:
            self._view._FlagAsMethod(function_name)
```

[docs]

```
    def RefreshView(self):
        """
        Refresh the STAAD view window.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.RefreshView()
        """
        self._view.RefreshView()
```

[docs]

```
    def ShowAllMembers(self):
        """
        Show all members in the STAAD view.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowAllMembers()
        """
        self._view.ShowAllMembers()
```

[docs]

```
    def HideAllMembers(self):
        """
        Hide all members in the STAAD view.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.HideAllMembers()
        """
        self._view.HideAllMembers()
```

[docs]

```
    def ZoomExtentsMainView(self):
        """
        Zoom to extents in the main STAAD view.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ZoomExtentsMainView()
        """
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def ShowMembers(self, NMembers, NaMemberNos):
        """
        Show specific members in the STAAD view.

        Parameters
        ----------
        NMembers : int
            Number of members to show.
        NaMemberNos : list of int
            List of member numbers to show.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowMembers(2, [1, 2])
        """
        safe_list = make_safe_array_long_input(NaMemberNos)
        lista_variant = make_variant_vt_ref(safe_list, automation.VT_ARRAY | au

        self._view.ShowAllMembers()
        self._view.HideAllMembers()
        self._geometry.ClearMemberSelection()
        self._view.ShowMembers(NMembers, lista_variant)
        self._view.ShowIsometric()
        self._view.ZoomExtentsMainView()
        self._view.RefreshView()
```

[docs]

```
    def HideMember(self, IDMember):
        """
        Hide a specific member in the STAAD view.

        Parameters
        ----------
        IDMember : int
            Member number to hide.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.HideMember(1)
        """
        self._view.HideMember(IDMember)
        self._view.RefreshView()
```

[docs]

```
    def HideMembers(self, NMembers, NaMemberNos):
        """
        Hide specific members in the STAAD view.

        Parameters
        ----------
        NMembers : int
```

```
            Number of members to hide.
        NaMemberNos : list of int
            List of member numbers to hide.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.HideMembers(2, [1, 2])
        """
        safe_list = make_safe_array_long_input(NaMemberNos)
        lista_variant = make_variant_vt_ref(safe_list, automation.VT_ARRAY | au

        self._view.HideMembers(NMembers, lista_variant)
        self._view.RefreshView()
```

[docs]

```
    def ShowBack(self):
        """
        Set the view to the back orientation.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowBack()
        """
        self._view.ShowBack()
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def ShowBottom(self):
        """
        Set the view to the bottom orientation.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowBottom()
        """
        self._view.ShowBottom()
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def ShowFront(self):
        """
        Set the view to the front orientation.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowFront()
        """
        self._view.ShowFront()
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def ShowIsometric(self):
        """
        Set the view to isometric orientation.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowIsometric()
        """
        self._view.ShowIsometric()
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def ShowLeft(self):
        """
        Set the view to the left orientation.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowLeft()
        """
        self._view.ShowLeft()
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def ShowPlan(self):
        """
        Set the view to the plan (top) orientation.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowPlan()
        """
        self._view.ShowPlan()
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def ShowRight(self):
        """
        Set the view to the right orientation.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowRight()
        """
        self._view.ShowRight()
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SpinLeft(self, Degrees):
        """
        Spin the view to the left by a specified number of degrees.

        Parameters
        ----------
        Degrees : float or int
            Number of degrees to spin left.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SpinLeft(15)
        """
        Degrees = float(Degrees)
        self._view.SpinLeft(Degrees)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
def SpinRight(self, Degrees):
    """
    Spin the view to the right by a specified number of degrees.

    Parameters
    ----------
    Degrees : float or int
        Number of degrees to spin right.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.View.SpinRight(15)
    """
    Degrees = float(Degrees)
    self._view.SpinRight(Degrees)
    self._view.RefreshView()
    self._view.ZoomExtentsMainView()
```

[docs]

```
def ZoomAll(self):
    """
    Zoom to show all objects in the STAAD view.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.View.ZoomAll()
    """
    self._view.ZoomAll()
```

[docs]

```
def GetApplicationDesktopSize(self):
    """
    Get the size of the application desktop.

    Returns
    -------
    tuple of int
        (width, height) of the application desktop.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> width, height = staad_obj.View.GetApplicationDesktopSize()
```

```
        """
        safe_n1 = make_safe_array_int(1)
        L = make_variant_vt_ref(safe_n1,  automation.VT_I4)

        safe_n2 = make_safe_array_int(1)
        W = make_variant_vt_ref(safe_n2,  automation.VT_I4)

        self._view.GetApplicationDesktopSize(L, W)

        L = L[0]
        W = W[0]

        return (L, W)
```

[docs]

```
    def SetWindowPosition(self, xTop, yTop, xWindow, yWindow):
        """
        Set the position and size of the STAAD application window.

        Parameters
        ----------
        xTop : int
            X coordinate of the top-left corner.
        yTop : int
            Y coordinate of the top-left corner.
        xWindow : int
            Width of the window.
        yWindow : int
            Height of the window.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetWindowPosition(100, 100, 800, 600)
        """
        self._view.SetWindowPosition(xTop, yTop, xWindow, yWindow)
```

[docs]

```
    def RotateUp (self, dDegrees: float):
        """
        Rotates the structure through Degrees about the Global X-Axis.

        Parameters
        ----------
        dDegrees : float
            Variable providing the degree of rotation.

        Examples
        -------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.RotateUp(30)
        """
        self._view.RotateUp(dDegrees)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def RotateDown (self, dDegrees: float):
        """
        Rotates the structure through Degrees about the Global X-Axis.

        Parameters
        ----------
        dDegrees : float
            Variable providing the degree of rotation.

        Examples
        -------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.RotateDown(30)
        """
        self._view.RotateDown(dDegrees)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def RotateLeft (self, dDegrees: float):
        """
        Rotates the structure through Degrees about the Global Y-Axis.

        Parameters
        ----------
        dDegrees : float
            Variable providing the degree of rotation.

        Examples
        -------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.RotateLeft(30)
        """
        self._view.RotateLeft(dDegrees)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
def RotateRight (self, dDegrees: float):
    """
    Rotates the structure through Degrees about the Global Y-Axis.

    Parameters
    ----------
    dDegrees : float
        Variable providing the degree of rotation.

    Examples
    -------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.View.RotateRight(30)
    """
    self._view.RotateRight(dDegrees)
    self._view.RefreshView()
    self._view.ZoomExtentsMainView()
```

[docs]

```
def CreateNewViewForSelections (self):
    """
    Creates a new view in new window for the selected objects displayed in t

    Examples
    -------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.View.CreateNewViewForSelections()
    """
    return self._view.CreateNewViewForSelections()
```

[docs]

```
def SetLabel (self, which: int, showFlag: bool):
    """
    Sets the label on the structure diagram on or off.

    Parameters
    ----------
    which : int
        Variable identifying the diagram type. It may be one of the followi
            +------+------------------------------------------+
            | ID   | Label Type                               |
            +======+==========================================+
            | 0    | Node number label                        |
            +------+------------------------------------------+
            | 1    | Member number label                      |
            +------+------------------------------------------+
            | 2     | Member property reference label            |
            +-------+--------------------------------------------+
            | 3     | Material property reference label          |
            +-------+--------------------------------------------+
            | 4     | Support label                              |
            +-------+--------------------------------------------+
            | 5     | Member release label                       |
            +-------+--------------------------------------------+
            | 6     | Member orientation label                   |
            +-------+--------------------------------------------+
            | 7     | Member section label                       |
            +-------+--------------------------------------------+
            | 8     | Load value label                           |
            +-------+--------------------------------------------+
            | 9     | Axes label                                 |
            +-------+--------------------------------------------+
            | 10    | Node position label                        |
            +-------+--------------------------------------------+
            | 11    | Member specification label                 |
            +-------+--------------------------------------------+
            | 12    | Member ends                                |
            +-------+--------------------------------------------+
            | 13    | Plate element number label                 |
            +-------+--------------------------------------------+
            | 14    | Plate element orientation label            |
            +-------+--------------------------------------------+
            | 15    | Solid element number label                 |
            +-------+--------------------------------------------+
            | 16    | Dimension label                            |
            +-------+--------------------------------------------+
            | 17    | Floor load label                           |
            +-------+--------------------------------------------+
            | 18    | Floor load distribution diagram label      |
            +-------+--------------------------------------------+
            | 19    | Wind load label                            |
            +-------+--------------------------------------------+
            | 20    | Wind load influence area diagram label     |
            +-------+--------------------------------------------+
            | 21    | Diagram Info                               |
            +-------+--------------------------------------------+
    showFlag : bool
        Variable to set label mode on (True) or off (False).


    Examples
    -------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.View.SetLabel(20, True)
    """
    return self._view.SetLabel(which, showFlag)
    self._view.RefreshView()
    self._view.ZoomExtentsMainView()
```

[docs]

```
def SetSectionView (self, plane: int, minVal: float, maxVal: float):
    """
    Creates a section view of the structure.

    Parameters
    ----------
    plane : int
        Variable identifying the section plane.  It may be one of the follow
            +-----+------------------+
            | ID  | Values for plane |
            +=====+==================+
            | 0   | XY Plane         |
            +-----+------------------+
            | 1   | YZ Plane         |
            +-----+------------------+
            | 2   | XZ Plane         |
            +-----+------------------+
    minVal : float
        Minimum range of the cutting plane.
    maxVal : float
        Maximum range of the cutting plane.

    Examples
    -------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.View.SetSectionView(1, 0.4, 0.6)
    """
    self._view.SetSectionView(plane, minVal, maxVal)
    self._view.RefreshView()
    self._view.ZoomExtentsMainView()
```

[docs]

```
def SetDiagramMode (self, which: int, showFlag: bool, refreshFlag: bool):
    """
    Sets the label on the structure diagram on or off.

    Parameters
    ----------
    which : int
        Variable identifying the diagram type. It may be one of the followir

        +-----+--------------------------------------------------------+
        | ID  | Diagram Type                                           |
        +=====+========================================================+
        | 0   | Load                                                   |
        +-----+--------------------------------------------------------+
        | 1   | Displacement                                           |
        +-----+--------------------------------------------------------+
        | 2   | MY                                                     |
        +-----+--------------------------------------------------------+
        | 3   | MZ                                                     |
        +-----+--------------------------------------------------------+
        | 4   | FY                                                     |
        +-----+--------------------------------------------------------+
        | 5   | FZ                                                     |
        +-----+--------------------------------------------------------+
        | 6   | AX                                                     |
        +-----+--------------------------------------------------------+
        | 7   | TR                                                     |
        +-----+--------------------------------------------------------+
        | 8   | Structure                                              |
        +-----+--------------------------------------------------------+
        | 9   | Full Section                                           |
        +-----+--------------------------------------------------------+
        | 10  | Section Outline                                        |
        +-----+--------------------------------------------------------+
        | 11  | Stress                                                 |
        +-----+--------------------------------------------------------+
        | 12  | Shrink                                                 |
        +-----+--------------------------------------------------------+
        | 13  | Perspective                                            |
        +-----+--------------------------------------------------------+
        | 14  | Hide Structure                                         |
        +-----+--------------------------------------------------------+
        | 15  | Fill Plates & Solids                                   |
        +-----+--------------------------------------------------------+
        | 16  | Hide Plates & Solids                                   |
        +-----+--------------------------------------------------------+
        | 18  | Hide Piping                                            |
        +-----+--------------------------------------------------------+
        | 19  | Sort Geometry                                          |
        +-----+--------------------------------------------------------+
        | 20  | Sort Nodes                                             |
        +-----+--------------------------------------------------------+
        | 21  | Plate Stress                                           |
        +-----+--------------------------------------------------------+
        | 22  | Solid Stress                                           |
        +-----+--------------------------------------------------------+
        | 23  | Mode Shape                                             |
        +-----+--------------------------------------------------------+
        | 24  | Stress Animation                                       |
        +-----+--------------------------------------------------------+
        | 25  | Plate reinforcement                                    |
        +-----+--------------------------------------------------------+
        | 26  | Deck Influence Diagram*                                |
        +-----+--------------------------------------------------------+
        | 27  | Deck Carriageways*                                     |
        +-----+--------------------------------------------------------+
        | 28  | Deck Triangulation*                                    |
        +-----+--------------------------------------------------------+
        | 29  | Deck Loads*                                            |
        +-----+--------------------------------------------------------+
        | 30  | Deck Vehicles*                                         |
        +-----+--------------------------------------------------------+
        |     | (*) Requires the STAAD.beava component                 |
        +-----+--------------------------------------------------------+
    showFlag : bool
        Variable to set label mode on (True) or off (False).
    refreshFlag : bool
        Variable (True or False). If True, STAAD.Pro viewing windows refresh

    Examples
    -------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.View.SetDiagramMode(1, True, True)
    """
    self._view.SetDiagramMode(which, showFlag, refreshFlag)
    self._view.RefreshView()
    self._view.ZoomExtentsMainView()
```

[docs]

```
    def SetNodeAnnotationMode (self, dFlag: bool, refreshFlag: bool):
        """
        Sets the node displacement annotation mode.  This function works only ir

        Parameters
        ----------
        dFlag : bool
            Variable controlling the annotation type. It may be one of the follo
                +-----+-------------------------+
                | ID  | Annotation Type         |
                +=====+=========================+
                | 1   | X Displacement          |
                +-----+-------------------------+
                | 2   | Y Displacement          |
                +-----+-------------------------+
                | 3   | Z Displacement          |
                +-----+-------------------------+
                | 4   | Resultant Displacement  |
                +-----+-------------------------+
        refreshFlag : bool
            Variable (True or False). If True, STAAD.Pro viewing windows refresh

        Examples
        -------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetNodeAnnotationMode(1, True)
        """
        self._view.SetNodeAnnotationMode(dFlag, refreshFlag)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SetReactionAnnotationMode (self, dFlag: bool, refreshFlag: bool):
        """
        Sets the node displacement annotation mode.  This function works only ir

        Parameters
        ----------
        dFlag : bool
            Variable controlling the annotation type. It may be one of the follo
                +-----+-----------------------+
                | ID  | Annotation Type       |
                +=====+=======================+
                | 1   | X Reaction            |
                +-----+-----------------------+
                | 2   | Y Reaction            |
                +-----+-----------------------+
                | 3   | Z Reaction            |
                +-----+-----------------------+
                | 4   | X Rotation            |
                +-----+-----------------------+
                | 5   | Y Rotation            |
                +-----+-----------------------+
                | 6   | Z Rotation            |
                +-----+-----------------------+
                | 7   | Reaction Value Only   |
                +-----+-----------------------+
        refreshFlag : bool
            Variable (True or False). If True, STAAD.Pro viewing windows refresh

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetReactionAnnotationMode(1, True)
        """
        self._view.SetReactionAnnotationMode(dFlag, refreshFlag)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def GetInterfaceMode (self):
        """
        This function returns the current visual mode in the STAAD.Pro environme

        Returns
        -------
        int
            Returns 0 if Pre-processor or modeling mode.
            Returns 1 if Post-processing mode.
            Returns 2 if Interactive design mode for STAAD.etc interoperability
            Returns 4 if Piping mode.
            Returns 5 if BEAVA (i.e., Bridge Deck ) mode.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.GetInterfaceMode()
        """
        return self._view.GetInterfaceMode()
```

[docs]

```
    def SetInterfaceMode (self, interfaceMode: int):
        """
        This function sets the current visual mode in the STAAD.Pro environment

        Parameters
        ----------
        interfaceMode : int
            Variable to set the current visual mode in STAAD.Pro environment. Fo
                +-----+--------------------------------+
                |ID   | Mode Type                      |
                +=====+================================+
                | 0   | Pre-processor or modeling mode |
                +-----+--------------------------------+
                | 1   | Physical modeling mode         |
                +-----+--------------------------------+
                | 2   | Building planner mode          |
                +-----+--------------------------------+
                | 3   | Piping mode                    |
                +-----+--------------------------------+
                | 5   | Post Processing mode           |
                +-----+--------------------------------+
                | 6   | FoundationDesign mode          |
                +-----+--------------------------------+
                | 7   | ConnectionDesign mode          |
                +-----+--------------------------------+
                | 9   | AdvancedConcreteDesign mode    |
                +-----+--------------------------------+
                | 10  | AdvancedSlabDesign mode        |
                +-----+--------------------------------+
                | 11  | Earthquake mode                |
                +-----+--------------------------------+
                | 12  | SteelAutoDrafter mode          |
                +-----+--------------------------------+
                | 13  | ChineseSteelDesign mode        |
                +-----+--------------------------------+

        Returns
        -------
        int
            Returns 0 if Pre-processor or modeling mode.
            Returns 1 if Post-processing mode.
            Returns 2 if Interactive design mode for STAAD.etc interoperability
            Returns 4 if Piping mode.
            Returns 5 if BEAVA (i.e., Bridge Deck ) mode.
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetInterfaceMode(1)
        """
        self._view.SetInterfaceMode(interfaceMode)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SetModeSectionPage (self, interfaceMode: int, sectionNumber: int, pageNu
        """
        This function sets the current page mode in the STAAD.Pro environment.

        Parameters
        ----------
        interfaceMode : int
            Variable to set the current visual mode in STAAD.Pro environment. Fo
                +----+------------------------------------------------------------
                | ID | Interface Mode
                +====+============================================================
                | 0  | Pre-processor or modeling mode
                +----+------------------------------------------------------------
                | 1  | Post-processing mode
                +----+------------------------------------------------------------
                | 2  | Interactive design mode for STAAD.etc interoperability
                +----+------------------------------------------------------------
                | 4  | Piping mode
                +----+------------------------------------------------------------
                | 5  | BEAVA (i.e., Bridge Deck) mode
                +----+------------------------------------------------------------
        sectionNumber : int
            Variable to set the current main page (the tabs on the left-hand sid
                +----+----------------------+
                | ID | Main Page            |
                +====+======================+
                | 1  | Setup page           |
                +----+----------------------+
                | 2  | Geometry page        |
                +----+----------------------+
                | 3  | General page         |
                +----+----------------------+
                | 5  | Node Results page    |
                +----+----------------------+
                | 6  | Beam Result page     |
                +----+----------------------+
                | 7  | Plate Results page   |
                +----+----------------------+
                | 8  | Solid Results page   |
                +----+----------------------+
        pageNumber : int
            Variable  to set the current sub page (within a particular main page
                +----+-------------------------------+
                | ID | Page Number                   |
                +====+===============================+
                | 0  | Job Info page                 |
                +----+-------------------------------+
                | 1  | Beam page                     |
                +----+-------------------------------+
                | 4  | Plate page                    |
                +----+-------------------------------+
                | 5  | Solid page                    |
                +----+-------------------------------+
                | 6  | Property page                 |
                +----+-------------------------------+
                | 7  | Constant page                 |
                +----+-------------------------------+
                | 8  | Material page                 |
                +----+-------------------------------+
                | 9  | Support page                  |
                +----+-------------------------------+
                | 10 | Member Specifications page    |
                +----+-------------------------------+
                | 11 | Load page                     |
                +----+-------------------------------+
                | 17 | Reaction page                 |
                +----+-------------------------------+
                | 18 | Displacement page             |
                +----+-------------------------------+
                | 19 | Failure page                  |
                +----+-------------------------------+
                | 20 | Forces page                   |
                +----+-------------------------------+
                | 21 | Beam Stress page              |
                +----+-------------------------------+
                | 22 | Plate Stress page             |
                +----+-------------------------------+
                | 23 | Solid Stress page             |
                +----+-------------------------------+


        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetModeSectionPage(1,6,20)
        """
        self._view.SetModeSectionPage(interfaceMode, sectionNumber, pageNumber)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SetBeamAnnotationMode (self, Type: int, DWFlags: int, RefreshFlag: bool
        """
        This function sets the current page mode in the STAAD.Pro environment.
        Parameters
        ----------
        Type : int
            Variable controlling the annotation type.  It may be one of the foll
                +----+----------------------+
                | ID | Annotation Type      |
                +====+======================+
                | 0  | Axial Diagram        |
                +----+----------------------+
                | 1  | Torsion Diagram      |
                +----+----------------------+
                | 2  | Moment Diagram       |
                +----+----------------------+
                | 3  | Shear Diagram        |
                +----+----------------------+
                | 4  | Stress Diagram       |
                +----+----------------------+
                | 5  | Displacement Diagram |
                +----+----------------------+
        DWFlags : int
            Variable controlling what values are to be shown for the annotation
                +----+-----------------------+
                | ID | Values                |
                +====+=======================+
                | 1  | End Values            |
                +----+-----------------------+
                | 2  | Max Absolute Values   |
                +----+-----------------------+
                | 3  | Mid-span Values       |
                +----+-----------------------+
        RefreshFlag : int
            Boolean variable (True or False). If True, STAAD.Pro viewing windows

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetBeamAnnotationMode(2, 1, True)
        """
        self._view.SetBeamAnnotationMode(Type, DWFlags, RefreshFlag)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def ShowMember (self, nMember: int):
        """
        Show the specified member.

        Parameters
        ----------
        nMember : int
            Variable that holds member number to be shown.
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ShowMember(5)
        """
        self._view.ShowMember(nMember)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SetUnits (self, uType: int, strUnit: str):
        """
        Set viewing unit for the active view.

        Parameters
        ----------
        uType : int
            Variable that holds unit type. Values are as follows:
                +-----+------------------------------+
                | ID  | Unit Type                    |
                +=====+==============================+
                | 0   |Dimension                     |
                +-----+------------------------------+
                | 1   |Displacement                  |
                +-----+------------------------------+
                | 2   |SectionDimension              |
                +-----+------------------------------+
                | 3   |SectionArea                   |
                +-----+------------------------------+
                | 4   |Inertia                       |
                +-----+------------------------------+
                | 5   |Force                         |
                +-----+------------------------------+
                | 6   |Moment                        |
                +-----+------------------------------+
                | 7   |DistributedForce              |
                +-----+------------------------------+
                | 8   |DistributedMoment             |
                +-----+------------------------------+
                | 9   |Density                       |
                +-----+------------------------------+
                | 10  |Acceleration                  |
                +-----+------------------------------+
                | 11  |Spring                        |
                +-----+------------------------------+
                | 12  |RotSpring                     |
                +-----+------------------------------+
                | 13  |MaterialModulus               |
                +-----+------------------------------+
                | 14  |Stress                        |
                +-----+------------------------------+
                | 15  |Alpha                        |
                +-----+-----------------------------+
                | 16  |Temperature                  |
                +-----+-----------------------------+
                | 17  |Mass                         |
                +-----+-----------------------------+
                | 18  |SectionModulus               |
                +-----+-----------------------------+
                | 19  |RotationalDisplacement       |
                +-----+-----------------------------+
                | 20  |SubgradeModulus              |
                +-----+-----------------------------+
                | -1  |NoUnit                       |
                +-----+-----------------------------+
        strUnit : str
            Variable array that holds the unit for the specified type. Like "cm'


        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetUnits(0, "cm")
        """
        self._view.SetUnits(uType, strUnit)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def HidePlate (self, nPlate: int):
        """
        Hide the specified plate.

        Parameters
        ----------
        nPlate : int
            Variable that holds plate number to be hidden.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.HidePlate(5)
        """
        self._view.HidePlate(nPlate)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def HideSolid (self, nSolid: int):
        """
        Hide the specified solid.

        Parameters
        ----------
        nSolid : int
            Variable that holds solid number to be hidden.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.HideSolid(5)
        """
        self._view.HideSolid(nSolid)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def HideSurface (self, nSurface: int):
        """
        Hide the specified surface.

        Parameters
        ----------
        nSurface : int
            Variable that holds surface number to be hidden.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.HideSurface(5)
        """
        self._view.HideSurface(nSurface)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def HideEntity (self, nEntity: int):
        """
        Hides the specified entity, which may be a Beam, Plate, Solid, or Surfac

        Parameters
        ----------
        nEntity : int
            Variable that holds an entity (i.e., Member, Plates etc.) number to

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.HideEntity(5)
        """
        self._view.HideEntity(nEntity)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SelectMembersParallelTo (self, bstrAxis: str):
        """
        Select members parallel to the specified axis.

        Parameters
        ----------
        bstrAxis : str
            Variable that holds the Axis ID. It may have three values:
                +----+----------+
                | ID | Axis     |
                +====+==========+
                | X  | X-Axis   |
                +----+----------+
                | Y  | Y-Axis   |
                +----+----------+
                | Z  | Z-Axis   |
                +----+----------+

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectMembersParallelTo(5)
        """
        self._view.SelectMembersParallelTo(bstrAxis)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SelectGroup (self, bstrGroup: str):
        """
        Select the relevant entities of the specified group.

        Parameters
        ----------
        bstrGroup : str
            A string variable that holds the group name.

        Returns
        -------
        int
```

```
            Returns True if successful
            Returns False if unsuccessful

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectGroup("grp_name")
        """
        return self._view.SelectGroup(bstrGroup)
```

[docs]

```
    def SelectInverse (self, entityType: int):
        """
        Inverse geometry selection for the specified entity.

        Parameters
        ----------
        entityType : int
            Variable that holds entity type. Values may be as follows:
                +-----+----------------+
                | ID  | Entity Type    |
                +=====+================+
                | 1   | Node           |
                +-----+----------------+
                | 2   | Beam           |
                +-----+----------------+
                | 3   | Plate          |
                +-----+----------------+
                | 4   | Solid          |
                +-----+----------------+
                | 5   | Surface        |
                +-----+----------------+

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectInverse(1)
        """
        self._view.SelectInverse(entityType)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SelectByItemList (self, entityType: int, nItems: int, itemList: list):
        """
        Select entities as specified.

        Parameters
        ----------
        entityType : int
            Variable that holds entity type. Values may be as follows:
                +-----+----------------+
                | ID  | Entity Type    |
                +=====+================+
                | 1   | Node           |
                +-----+----------------+
                | 2   | Beam           |
                +-----+----------------+
                | 3   | Plate          |
                +-----+----------------+
                | 4   | Solid          |
                +-----+----------------+
                | 5   | Surface        |
                +-----+----------------+
        nItems : int
            Variable that holds total number of entities needs to be selected.
        itemList : list of int
            List holds the entity nos, which need to be selected.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectByItemList(1, 2, [1, 2])
        """
        entityList = make_safe_array_long_input(itemList)
        self._view.SelectByItemList(entityType, nItems, entityList)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SelectByMissingAttribute (self, attributeCode: int):
        """
        Select entity list for which specified entity is missing.

        Parameters
        ----------
        attributeCode : int
            Variable that holds attribute type. Values may be as follows:
                +----+--------------------------------------------------------------+
                | ID | Entity Type                                                  |
                +====+==============================================================+
                | 1  | Missing Property                                             |
                +----+--------------------------------------------------------------+
                | 2  | Missing Modulus of Elasticity                                |
                +----+--------------------------------------------------------------+
                | 3  | Missing Density of Material                                  |
                +----+--------------------------------------------------------------+
                | 4  | Missing Alpha (Coefficient of Thermal Expansion)             |
                +----+--------------------------------------------------------------+
                | 5  | Missing Poisson Ratio                                        |
                +----+-------------------------------------------------------------+

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectByMissingAttribute(5)
        """
        self._view.SelectByMissingAttribute(attributeCode)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SelectEntitiesConnectedToNode (self, entityType: int, nodeNo: int):
        """
        Select entities as specified in type and connected with the specified no

        Parameters
        ----------
        entityType : int
            Variable that holds entity type. Values may be as follows:
                +-----+----------------+
                | ID  | Entity Type    |
                +=====+================+
                | 0   | Geometry       |
                +-----+----------------+
                | 1   | Beam           |
                +-----+----------------+
                | 2   | Plate          |
                +-----+----------------+
                | 3   | Solid          |
                +-----+----------------+
        nodeNo : int
            Variable that holds node numbers with which connected entities needs

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectEntitiesConnectedToNode(0, 1)
        """
        self._view.SelectEntitiesConnectedToNode(entityType, nodeNo)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SelectEntitiesConnectedToMember (self, entityType: int, memberNo: int):
        """
        Select entities as specified in type and connected with the specified Me
        Parameters
        ----------
        entityType : int
            Variable that holds entity type. Values may be as follows:
                +-----+----------------+
                | ID  | Entity Type    |
                +=====+================+
                | 0   | Geometry       |
                +-----+----------------+
                | 1   | Beam           |
                +-----+----------------+
                | 2   | Plate          |
                +-----+----------------+
                | 3   | Solid          |
                +-----+----------------+
        memberNo : int
            Variable that holds Member numbers with which connected entities nee

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectEntitiesConnectedToMember(1, 2)
        """
        self._view.SelectEntitiesConnectedToMember(entityType, memberNo)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SelectEntitiesConnectedToPlate (self, entityType: int, plateNo: int):
        """
        Select entities as specified in type and connected with the specified Pl

        Parameters
        ----------
        entityType : int
            Variable that holds entity type. Values may be as follows:
                +-----+----------------+
                | ID  | Entity Type    |
                +=====+================+
                | 0   | Geometry       |
                +-----+----------------+
                | 1   | Beam           |
                +-----+----------------+
                | 2   | Plate          |
                +-----+----------------+
                | 3   | Solid          |
                +-----+----------------+
        plateNo : int
            Variable that holds Plate numbers with which connected entities need

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectEntitiesConnectedToPlate(2, 3)
        """
        self._view.SelectEntitiesConnectedToPlate(entityType, plateNo)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def SelectEntitiesConnectedToSolid (self, entityType: int, solidNo: int):
        """
        Select entities as specified in type and connected with the specified So

        Parameters
        ----------
        entityType : int
            Variable that holds entity type. Values may be as follows:
                +-----+----------------+
                | ID  | Entity Type    |
                +=====+================+
                | 0   | Geometry       |
                +-----+----------------+
                | 1   | Beam           |
                +-----+----------------+
                | 2   | Plate          |
                +-----+----------------+
                | 3   | Solid          |
                +-----+----------------+
        solidNo : int
            Variable that holds Solid numbers with which connected entities need

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SelectEntitiesConnectedToSolid(3, 4)
        """
        self._view.SelectEntitiesConnectedToSolid(entityType, solidNo)
        self._view.RefreshView()
        self._view.ZoomExtentsMainView()
```

[docs]

```
    def GetNoOfBeamsInView (self):
        """
        Get No Of Beams In View

        Returns
        -------
        int
```

```
            Returns number of beams present in view.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.GetNoOfBeamsInView()
        """
        return self._view.GetNoOfBeamsInView()
```

[docs]

```
    def GetBeamsInView (self, nBeamList: list):
        """
        Get Beams In View

        Parameters
        ----------
        nBeamList : nBeamList
            Collection of beam

        Returns
        -------
        int
            Returns number of beams present in view.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.GetBeamsInView([1, 2, 4])
        """
        nBeamList_ref = make_safe_array_long_input(nBeamList)
        return self._view.GetBeamsInView(nBeamList_ref)
```

[docs]

```
    def CreateNewViewForSelectionsEx (self, windowOptions: int):
        """
        Creates a new view for the selected objects displayed in the active wind

        Parameters
        ----------
        windowOptions : int
            0 = Creates a new window for the view, 1 = Display the view in the a

        Returns
        -------
        bool
            Returns True Creation of new view is successful.
            Returns False Creation of new view is unsuccessful.
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.CreateNewViewForSelectionsEx(1)
        """
        return self._view.CreateNewViewForSelectionsEx(windowOptions)
```

[docs]

```
    def ExportView (self, FileLocation: str, FileName: str, FileFormat: int, Ov
        """
        Used for exporting the information displayed in the active view window i

        Parameters
        ----------
        FileLocation : str
            Location of the saved view file (Folder need to be present otherwise
        FileName : str
            Name of the saved view file.
        FileFormat : int
            0 = bmp, 1 = jpg, 2 = tga, 3 = tif - Create the view in the specifi
        Overwrite : bool
            Boolean for provide option to  overwrite an existing file.
                - True - Allow Overwrite
                - False - No overwrite

        Returns
        -------
        int
            Returns 1 if Export view is successful.
            Returns -1 if Generic Error.
            Returns -100 if Invalid Argument.
            Returns -1003 if File already exist.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.ExportView(r"<folderPath>", "<fileName>", 1, True)
        """
        return self._view.ExportView(FileLocation, FileName, FileFormat, Overwr
```

[docs]

```
    def CopyPicture (self):
        """
        Copy active view to clipbord and gives size of image in it's referance v

        Returns
        -------
        Tuple
```

```
            Returns a tuple containing xDim size of image in x direction (Length

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.CopyPicture()
        """
        safe_xDim = make_safe_array_long(0)
        xDim = make_variant_vt_ref(safe_xDim,  automation.VT_I4)
        safe_yDim = make_safe_array_long(0)
        yDim = make_variant_vt_ref(safe_yDim,  automation.VT_I4)

        self._view.CopyPicture(xDim, yDim)
        return (xDim[0], yDim[0])
```

[docs]

```
    def GetScaleValues (self):
        """
        Obtain the current set of scales used for displaying loads and results s

        Returns
        -------
        list of float
            Returns list of float type and size same as number of scale types. A
                +----+---------------+-----------------+---------------------+
                | ID | Type          | Scale Items     | Unit per length     |
                +====+===============+=================+=====================+
                | 0  | Loads         | Point Force     | Force               |
                +----+---------------+-----------------+---------------------+
                | 1  | Loads         | Dist. Force     | Force/length        |
                +----+---------------+-----------------+---------------------+
                | 2  | Loads         | Point Moment    | Force*length        |
                +----+---------------+-----------------+---------------------+
                | 3  | Loads         | Dist. Moment    | Force*length/length |
                +----+---------------+-----------------+---------------------+
                | 4  | Loads         | Pressure        | Force/length^2      |
                +----+---------------+-----------------+---------------------+
                | 5  | Results       | Bending Y       | Force*length        |
                +----+---------------+-----------------+---------------------+
                | 6  | Results       | Bending Z       | Force*length        |
                +----+---------------+-----------------+---------------------+
                | 7  | Results       | Shear Y         | Force               |
                +----+---------------+-----------------+---------------------+
                | 8  | Results       | Shear Z         | Force               |
                +----+---------------+-----------------+---------------------+
                | 9  | Results       | Axial           | Force               |
                +----+---------------+-----------------+---------------------+
                | 10 | Results       | Torsion         | Force*length        |
                +----+---------------+-----------------+---------------------+
                | 11 | Results       | Displacement    | Length              |
                +----+---------------+-----------------+---------------------+
                | 12 | Results       | Beam Stress     | Force/length^2       |
                +----+---------------+-----------------+----------------------+
                | 13 | Results       | Mode Shape      | (none)               |
                +----+---------------+-----------------+----------------------+

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> list = staad_obj.View.GetScaleValues()
        >>> print(list)
        """
        scaleCount = self._view.GetScaleCount()
        scale_safe_list = make_safe_array_double(scaleCount)
        scaleList = make_variant_vt_ref(scale_safe_list,  automation.VT_ARRAY |
        self._view.GetScaleValues(scaleList)
        return scaleList[0]
```

[docs]

```
def SetScaleValues (self, ScalesList: list):
    """
    Set the scales used for displaying loads and results as shown in the Dia

    Parameters
    ----------
    ScalesList : list of float
        List of float type and size same as number of scale types. API sets

    Returns
    -------
    int
        Returns 1 if Values were successfully updated.
        Returns 0 if Values could not be updated.
        Returns -1 if Error with defined array.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.View.SetScaleValues([1.0, 2.5, 3.2])
    """
    ScalesList_safe = make_safe_array_double_input(ScalesList)
    return self._view.SetScaleValues(ScalesList_safe)
```

[docs]

```
def GetScaleValueByType (self, scaleTypeId: int):
    """
    Obtain the value of the scale that is used to display a specified load o

    Parameters
```

```
        ----------
        scaleTypeId : int
            The index of the required load or result type

        Returns
        -------
        float
            Returns value of scale type listed, in Base Units

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> value = staad_obj.View.GetScaleValueByType(1)
        >>> print(value)
        """
        safe_value = make_safe_array_double(0)
        value = make_variant_vt_ref(safe_value,  automation.VT_R8)

        self._view.GetScaleValueByType(scaleTypeId, value)
        return value[0]
```

[docs]

```
    def SetScaleValueByType (self, scaleTypeId: int, value: float):
        """
        Set the scale used for displaying a chosen load or result diagram as sho

        Parameters
        ----------
        scaleTypeId : int
            The index of the required load or result type to be set.
        value : float
            Value of scale type to be used.

        Returns
        -------
        bool
            Returns 1/TRUE Value was successfully updated.
            Returns 0/FALSE Value could not be updated.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> retValue = staad_obj.View.SetScaleValueByType(1, 1.2)
        """
        return self._view.SetScaleValueByType(scaleTypeId, value)
```

[docs]

```
    def GetScaleCount (self):
        """
        Returns the count of scales that are used in STAAD.Pro which can be reac

        Returns
        -------
        int
            Returns Positive_Value Count of scales.
            Returns 0/Negative_Value Unsuccessful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> scaleCount = staad_obj.View.GetScaleCount()
        """
        return self._view.GetScaleCount()
```

[docs]

```
    def DetachView (self):
        """
        Remove a view from the collection of saved views. The view to be removed

        Returns
        -------
        int
            Returns 1 if View successfully detached.
            Returns 0 if Unsuccessful
            Returns -1 if Generic Error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.View.DetachView()
        """
        return self._view.DetachView()
```

[docs]

```
    def RenameView (self, viewName: str):
        """
        Renames a saved view. The view should be open and be the active window.

        Parameters
        ----------
        viewName : str
            New name of the saved view

        Returns
        -------
        int
```

```
            Returns 1 if Rename view is successful.
            Returns 0 if Unsuccessful
            Returns 2 if View name already used.
            Returns -1 if Generic Error
            Returns -100 if Invalid Argument

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.View.RenameView("view1")
        """
        return self._view.RenameView(viewName)
```

[docs]

```
    def OpenView (self, viewName: str, windowOptions: bool):
        """
        Open a previously saved view in either the active window or create a nev

        Parameters
        ----------
        viewName : str
            New name of the saved view
        windowOptions : bool
            False = Creates a new window for the view which becomes the active v

        Returns
        -------
        int
            Returns 1 if View Successfully opened.
            Returns 0 if Unsuccessful
            Returns 2 if View name does not exist.
            Returns -1 if Generic Error
            Returns -100 if Invalid Argument

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.View.OpenView("view1", True)
        >>> print(status)
        """
        return self._view.OpenView(viewName, windowOptions)
```

[docs]

```
    def SaveView (self, viewName: str, overWrite: bool):
        """
        Save the active graphic view to the collection of saved views which can

        Parameters
        ----------
        viewName : str
            New name for the view
        overWrite : bool
            Option to overwrite if the given viewName already exists. False = Do

        Returns
        -------
        int
            Returns 1 if Save view is successful.
            Returns 0 if Unsuccessful
            Returns 2 if View name already exist and overWrite is false.
            Returns -1 if Generic Error
            Returns -100 if Invalid Argument

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.View.SaveView("view1", True)
        >>> print(status)
        """
        return self._view.SaveView(viewName, overWrite)
```

[docs]

```
    def GetWindowTitle (self, id: int):
        """
        Returns the Title of the Window.

        Parameters
        ----------
        id : int
            The index of the required Window (Type: Long). Note that IDs start f

        Returns
        -------
        str
            Returns the Window string title.
            Returns Empty_String Window id not found.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> title = staad_obj.View.GetWindowTitle(1)
        """
        return self._view.GetWindowTitle(id)
```

[docs]

```
    def GetWindowCount (self):
        """
        Get the number of windows currently open. This includes both graphic wir

        Returns
        -------
        int
            Returns Positive_Number The count of open Window.
            Returns -1 if Error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.View.GetWindowCount()
        """
        return self._view.GetWindowCount()
```

[docs]

```
    def CloseActiveWindow (self):
        """
        Closes the active graphic or table window, however there must be at leas

        Returns
        -------
        bool
            Returns True if Window closed.
            Returns FALSE if Unsuccessful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.View.CloseActiveWindow()
        """
        return self._view.CloseActiveWindow()
```

[docs]

```
    def SetActiveWindow (self, id: int):
        """
        Set a given window (active graphic or table window) with the provided ic

        Parameters
        ----------
        id : int
            The id of the window to be made the active window.

        Returns
        -------
        bool
            Returns True if successful.
            Returns FALSE if unsuccessful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetActiveWindow(2)
        """
        return self._view.SetActiveWindow(id)
```

[docs]

```
    def SetDesignResults (self, utilization: int, color: bool, showValues: bool
        """
        Sets Design Results to active view, this function replicates the setting

        Parameters
        ----------
        ld : int
            Value of type Long. (0 = None, 1 = Actual Ratio, 2 = Normalised Rati
        color : bool
            Value of type Boolean. (False/0 = Basic Colored, True/1 = Detailed C
        showValues : bool
            Value of type Boolean. (False/0 = Do Not Show Values, True/1 = Show

        Returns
        -------
        int
            Returns 1 if Set Design Results is successful.
            Returns 0 if Unsuccessful.
            Returns -1 if Generic Error.
            Returns -2 if Design Results not Loaded.
            Returns -100 if Invalid Argument.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.View.SetDesignResults(1, True, True)
        """
        return self._view.SetDesignResults(utilization, color, showValues)
```