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

[docs]

```
class OSLoad:
    CoInitialize()
```

[docs]

```
    def __init__(self, staadObj):
        self._staad = staadObj
        self._load = self._staad.Load

        self._functions= [
            'CreateNewPrimaryLoad',
            'CreateNewLoadCombination',
            'CreateNewReferenceLoad',
            'CreateLoadEnvelop',
            'CreateLoadList',
            'CreateNewPrimaryLoadEx',
            'CreateNewPrimaryLoadEx2',
            'SetLoadActive',
            'SetReferenceLoadActive',
            'SetLoadType',
            'SetASDLoadAttribute',
            'SetLSDLoadAttribute',
            'AddSelfWeightInXYZ',
            'AddSelfWeightInXYZToGeometry',
            'AddNodalLoad',
            "AddSupportDisplacement",
            "AddMemberUniformForce",
            "AddMemberUniformMoment",
            'AddMemberConcForce',
            'AddMemberConcMoment',
            'AddMemberLinearVari',
            'AddMemberTrapezoidal',
            'AddMemberAreaLoad',
            'AddMemberFixedEnd',
            'AddElementPressure',
            'AddElementPressure',
            'AddElementHydrostaticPressure',
            'AddTemperatureLoad',
            'AddStrainLoad',
            'AddLoadAndFactorToCombination',
            'AddMemberFloorLoad',
            'AddMemberFloorLoadEx',
            'AddElementTrapPressureEx',
            'AddWindDefinition',
            'AddWindIntensity',
            'AddWindExposure',
            'AddWindLoad',
            'AddSeismicDefinition',
            'AddSeismicDefSelfWeight',
            'AddSeismicDefMemberWeight',
            'AddSeismicDefJointWeight',
            'AddSeismicDefElementWeight',
            'AddSeismicDefFloorWeight',
            'AddSeismicLoad',
            'AddAutoLoadCombinations',
            'AddRepeatLoad',
            'AddLoadCasesToEnvelop',
            'AddReferenceLoad',
            'AddSeismicDefWallArea',
            'AddWindDefinitionASCE7Parameters',
            'AddNotionalLoad',
            'AddDirectAnalysisDefinitionParameter',
            'AddResponseSpectrumLoadEx',
            'AddAutoCombinationRepeat',
            'RemoveLoadCasesFromEnvelop',
            'RemoveAttribute',
            'ClearPrimaryLoadCase',
            'ClearReferenceLoadCase',
            'IsDynamicLoadIncluded',
            'IsCombinationCase',
            'SplitLoadsOnBeam',
            'MergeLoadsOnBeam',
            'BeginLoadMerging',
            'EndLoadMerging',
            'ModifySeismicDefinitionParams',
            'ComputeWallWindPressureProfile',
            'ComputeWallWindPressureProfileASCE72016',
            'DeleteLoadEnvelop',
            'DeleteLoadList',
            'DeletePrimaryLoadCases',
            'DeleteReferenceLoadCases',
            'DeleteWindDefinition',
            'DeleteDirectAnalysisDefinitionParameter',
            'DeleteDirectAnalysisDefinition',
            'GetPrimaryLoadCaseCount',
            'GetPrimaryLoadCaseNumbers',
            'GetLoadCombinationCaseCount',
            'GetLoadCombinationCaseNumbers',
            'GetReferenceLoadCount',
            'GetReferenceLoadCaseCount',
            'GetReferenceLoadCaseNumbers',
            'GetNoOfSetsInReferenceLoad',
            'GetReferenceLoadByIndex',
            'GetReferenceLoadType',
            'GetReferenceLoadCaseTitle',
            'GetBeamCountAtFloor',
            'GetInfluenceArea',
            'GetActiveLoad',
            'GetNodalLoadCount',
            'GetNodalLoads',
            'GetUDLLoadCount',
            'GetUDLLoads',
            'GetUNIMomentCount',
            'GetUNIMoments',
            'GetTrapLoadCount',
            'GetTrapLoads',
            'GetConcForceCount',
            'GetConcForces',
            'GetConcMomentCount',
            'GetConcMoments',
            'GetNoOfLoadAndFactorPairsForCombination',
            'GetLoadAndFactorForCombination',
            'GetLoadCaseTitle',
            'GetElementPressureLoadCount',
            'GetElementPressureLoads',
            'GetElementConcLoadCount',
            'GetElementConcLoads',
            'GetLoadType',
            'GetLoadListCount',
            'GetLoadCountInLoadList',
            'GetLoadsInLoadList',
            'GetAttribute',
            'GetLoadType',
            'GetRepeatLoadCount',
            'GetNoLoadFactorInRepeatLoad',
            'GetRepeatLoadByIndex',
            'GetLinearVaryingLoadCount',
            'GetLinearVaryingLoads',
            'GetLoadTypeCount',
            'GetListSizeForLoadType',
            'GetAssignmentListForLoadType',
            'GetNodalLoadInfo',
            'GetMemberLoadInfo',
            'GetElementLoadInfo',
            'GetNotionalLoadCount',
            'GetNoLoadFactorDirectionInNotionalLoad',
            'GetNotionalLoadByIndex',
            'GetLoadItemsCount',
            'GetLoadItemType',
            'GetEnvelopeCount',
            'GetLoadEnvelopeDetails',
            'GetLoadListfromLoadEnvelope',
            'GetEnvelopeIDs'
        ]

        for function_name in self._functions:
            self._load._FlagAsMethod(function_name)



    ## SUPPORT FUNCTIONS


```

[docs]

```
    def CreateNewPrimaryLoad(self, primaryLoadTitle:str):
        """
        Creates a new PRIMARY load case.

        Parameters
        ----------
        primaryLoadTitle : str
            Title of the primary load case.

        Returns
        -------
        int
            Load number ID if the load case is created successfully.
            Returns -1 in case of a general error.
            Returns -8004 if the load case creation fails specifically.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> load_id = staad_obj.load.CreateNewPrimaryLoad("Dead Load")
        >>> print(load_id)
        """

        return self._load.CreateNewPrimaryLoad(primaryLoadTitle)
```

[docs]

```
    def CreateNewLoadCombination(self, loadCombTitle:str, loadCombNo:int):

        """
        Creates a new load combination case.

        Parameters
        ----------
        loadCombTitle : str
            Title of the load combination.
        loadCombNo : int
            Load combination number.

        Returns
        -------
        int
            Load number ID assigned to the load combination.
            Returns -1 in case of an error.
            Returns -8004 if it fails to create the load.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> comb_id = staad_obj.Load.CreateNewLoadCombination("DL+LL", 2)
        >>> print(comb_id)
        """
        return self._load.CreateNewLoadCombination(loadCombTitle, loadCombNo)
```

[docs]

```
    def CreateNewReferenceLoad(self, nodeNo:int, referenceLoadCaseTitle:str, lo
        """
        Creates a new reference load case.

        Parameters
        ----------
        nodeNo : int
            Reference ID to be assigned to the new reference load case.
        referenceLoadCaseTitle : str
            Title of the reference load case.
        loadType : int
            Type of load.

        Returns
        -------
        int
            Reference load case number ID.
            Returns -1 in case of an error.
            Returns -8004 if it fails to create the load.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ref_id = staad_obj.Load.CreateNewReferenceLoad(1, "Ref Load", 0)
        >>> print(ref_id)
        """
        return self._load.CreateNewReferenceLoad(nodeNo, referenceLoadCaseTitle
```

[docs]

```
    def CreateLoadEnvelop(self , envelopNumber:int,  envelopType:int, loadCaseL
        """
        Creates a Load Envelop with specified primary load case(s) and envelop t

        Parameters
        ----------
        envelopNumber : int
            Load Envelop reference ID
        envelopType : int
            Type of the load envelop:
                +-------+------------------+
                | Value |Load Envelop Type |
                +=======+==================+
                | 0     |NONE              |
                +-------+------------------+
                | 1     |STRESS            |
                +-------+------------------+
                | 2     |SERVICEABILITY    |
                +-------+------------------+
                | 3     |COLUMN            |
                +-------+------------------+
                | 4     |CONNECTION        |
                +-------+------------------+
                | 5     |STRENGTH          |
                +-------+------------------+
                | 6     |TEMPORARY         |
                +-------+------------------+
        loadCaseList : list of int
            Load Case IDs for which to create a load envelop

        Returns
        -------
        bool
            True OK.
            False General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.CreateLoadEnvelop(1, 1, [1,2,3])
        >>> print(result)
        """
        safe_LoadCaseList = make_safe_array_long_input(loadCaseList)
        return self._load.CreateLoadEnvelop(envelopNumber,  envelopType, safe_L
```

[docs]

```
    def CreateLoadList(self , listType:int, loadCaseList:list[int]):
        """
        Creates a load list.

        Parameters
        ----------
        listType : int
            Load list type: 0 and 1 for load list and load envelope list, respe
        loadCaseList : list of int
            Load Case reference IDs for which to create a load envelop

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.CreateLoadList(0, [1,2])
        >>> print(result)
        """
        safe_LoadCaseList = make_safe_array_long_input(loadCaseList)
        retval = self._load.CreateLoadList( listType, safe_LoadCaseList)


        # return bool(retval)
```

[docs]

```
    def CreateNewPrimaryLoadEx(self , primaryLoadTitle:str, loadType:int):
        """
        Creates new PRIMARY load case.

        Parameters
        ----------
        primaryLoadTitle : string
            The primary load case string title.
        loadType : int
            Type of the load:
                +-------+-----------+-------+-------------+
                | Value | Load Type | Value | Load Type   |
                +=======+===========+=======+=============+
                | 0     | Dead      | 12    | Traffic     |
                +-------+-----------+-------+-------------+
                | 1     | Live      | 13    | Temp        |
                +-------+-----------+-------+-------------+
                | 2     | Roof Live | 14    | Imperfection|
                +-------+-----------+-------+-------------+
                | 3     | Wind      | 15    | Accidental  |
                +-------+-----------+-------+-------------+
                | 4     | Seismic-H | 16    | Flood       |
                +-------+-----------+-------+-------------+
                | 5     | Seismic-V | 17    | Ice         |
                +-------+-----------+-------+-------------+
                | 6     | Snow      | 18    | Wind Ice    |
                +-------+-----------+-------+-------------+
                | 7     | Fluids    | 19    | Crane Hook  |
                +-------+-----------+-------+-------------+
                | 8     | Soil      | 20    | Mass        |
                +-------+-----------+-------+-------------+
                | 9     | Rain      | 21    | Gravity     |
                +-------+-----------+-------+-------------+
                | 10    | Ponding   | 22    | Push        |
                +-------+-----------+-------+-------------+
                | 11    | Dust      | 23    | None        |
                +-------+-----------+-------+-------------+

        Returns
        -------
        int
            Returns load Number of newly created Primary load Case.
            Returns -1 if general error.
            Returns -8004 if fail to create load.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> load_id = staad_obj.Load.CreateNewPrimaryLoadEx("Live Load", 1)
        >>> print(load_id)
        """
        return self._load.CreateNewPrimaryLoadEx( primaryLoadTitle, loadType)
```

[docs]

```
    def CreateNewPrimaryLoadEx2(self , primaryLoadTitle:str, loadType:int, loadC
        """
        Creates new PRIMARY load case.

        Parameters
        ----------
        primaryLoadTitle : string
            The primary load case string title.
        loadType : int
            Type of the load:
                +-------+-----------+-------+-------------+
                | Value | Load Type | Value | Load Type   |
                +=======+===========+=======+=============+
                | 0     | Dead      | 12    | Traffic     |
                +-------+-----------+-------+-------------+
                | 1     | Live      | 13    | Temp        |
                +-------+-----------+-------+-------------+
                | 2     | Roof Live | 14    | Imperfection|
                +-------+-----------+-------+-------------+
                | 3     | Wind      | 15    | Accidental  |
                +-------+-----------+-------+-------------+
                | 4     | Seismic-H | 16    | Flood       |
                +-------+-----------+-------+-------------+
                | 5     | Seismic-V | 17    | Ice         |
                +-------+-----------+-------+-------------+
                | 6     | Snow      | 18    | Wind Ice    |
                +-------+-----------+-------+-------------+
                | 7     | Fluids    | 19    | Crane Hook  |
                +-------+-----------+-------+-------------+
                | 8     | Soil      | 20    | Mass        |
                +-------+-----------+-------+-------------+
                | 9     | Rain      | 21    | Gravity     |
                +-------+-----------+-------+-------------+
                | 10    | Ponding   | 22    | Push        |
                +-------+-----------+-------+-------------+
                | 11    | Dust      | 23    | None        |
                +-------+-----------+-------+-------------+

        loadCaseNo : int
            The load case number.

        Returns
        -------
        int
            Returns load Case number of newly created Primary load Case.
            Returns 0 if not successfully
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> load_id = staad_obj.Load.CreateNewPrimaryLoadEx2("Wind", 3, 5)
        >>> print(load_id)
        """
        return self._load.CreateNewPrimaryLoadEx2( primaryLoadTitle, loadType, 
```

[docs]

```
    def SetLoadActive(self, loadNumber : int):
        """
        Activates the specified load number to allow adding or removing load ite

        Parameters
        ----------
        loadNumber : int
            Load case reference number ID.

        Returns
        -------
        bool
            True if the load case was successfully activated.
            False if an error occurred.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.SetLoadActive(1)
        >>> print(result)
        """
        return self._load.SetLoadActive(loadNumber)
```

[docs]

```
    def SetReferenceLoadActive(self, nLoadCaseNo : int):
        """
        Activates a reference load case to allow operations on its items.

        Parameters
        ----------
        nLoadCaseNo : int
            Reference load case ID in Load Case Details.

        Returns
        -------
        int
            Reference load case number ID.
            Returns -1 in case of an error.
            Returns -8002 if the load case is not found.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.SetReferenceLoadActive(2)
        >>> print(result)
        """
        return self._load.SetReferenceLoadActive(nLoadCaseNo)
```

[docs]

```
    def SetLoadType(self, loadCaseNumber : int, loadType:int):
        """
        Set load type to load case for considering load combination.

        Parameters
        ----------
        loadCaseNumber : int
            The load case reference number ID.
        loadType : int
            Type of the load.:
                +-------+-----------+-------+-------------+
                | Value | Load Type | Value | Load Type   |
                +=======+===========+=======+=============+
                | 0     | Dead      | 12    | Traffic     |
                +-------+-----------+-------+-------------+
                | 1     | Live      | 13    | Temp        |
                +-------+-----------+-------+-------------+
                | 2     | Roof Live | 14    | Imperfection|
                +-------+-----------+-------+-------------+
                | 3     | Wind      | 15    | Accidental  |
                +-------+-----------+-------+-------------+
                | 4     | Seismic-H | 16    | Flood       |
                +-------+-----------+-------+-------------+
                | 5     | Seismic-V | 17    | Ice         |
                +-------+-----------+-------+-------------+
                | 6     | Snow      | 18    | Wind Ice    |
                +-------+-----------+-------+-------------+
                | 7     | Fluids    | 19    | Crane Hook  |
                +-------+-----------+-------+-------------+
                | 8     | Soil      | 20    | Mass        |
                +-------+-----------+-------+-------------+
                | 9     | Rain      | 21    | Gravity     |
                +-------+-----------+-------+-------------+
                | 10    | Ponding   | 22    | Push        |
                +-------+-----------+-------+-------------+
                | 11    | Dust      | 23    | None        |
                +-------+-----------+-------+-------------+

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

```
            Returns 0 OK.
            Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.SetLoadType(1, 0)
        >>> print(result)
        """
        return self._load.SetLoadType(loadCaseNumber, loadType)
```

[docs]

```
    def SetASDLoadAttribute(self, loadCaseRefID : int, strengthType : int, allo
        """
        Sets Allowable Stress Design (ASD) load attribute.

        Parameters
        ----------
        loadCaseRefID : int
            Load case reference ID.
        strengthType : int
            Strength Type :
                +------------------------------------------------------+--------
                | Value                                                 | Integer
                +======================================================+========
                | STRENGTH_TYPE_NONE                                    | 0
                +------------------------------------------------------+--------
                | NORMAL_ASD_WORKING_STRESS_FORCES_WITHOUT_P_DELTA      | 1
                +------------------------------------------------------+--------
                | NORMAL_ASD_WORKING_STRESS_FORCES_WITH_P_DELTA         | 2
                +------------------------------------------------------+--------
                | STRENGTH_TYPE_OF_FORCES_WITHOUT_P_DELTA               | 3
                +------------------------------------------------------+--------
                | STRENGTH_TYPE_OF_FORCES_WITH_P_DELTA                  | 4
                +------------------------------------------------------+--------
                | COLUMN_ONLY_STRENGTH_TYPE_OF_FORCES_WITHOUT_P_DELTA   | 5
                +------------------------------------------------------+--------
                | COLUMN_ONLY_STRENGTH_TYPE_OF_FORCES_WITH_P_DELTA      | 6
                +------------------------------------------------------+--------

        allowStressIncrease : bool
            Allow 1/3 stress increase in ASD.

        Returns
        -------
        int
            Returns 0 OK.
            Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.SetASDLoadAttribute(1, 1, True)
        >>> print(result)
        """
        return self._load.SetASDLoadAttribute(loadCaseRefID, strengthType, allo
```

[docs]

```
    def SetLSDLoadAttribute(self, loadCaseRefID : int ):
        """
        Sets Limit State Design (LSD) load attribute.

        Parameters
        ----------
        loadCaseRefID : int
            Load case reference ID.

        Returns
        -------
        int
            Returns 0 OK.
            Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.SetLSDLoadAttribute(1)
        >>> print(result)
        """
        return self._load.SetLSDLoadAttribute(loadCaseRefID)
```

[docs]

```
    def AddSelfWeightInXYZ(self, varInDirection: int, varLoadFactor: float):
        """
        Adds self-weight to the active load case for all entities (beams, plates

        Parameters
        ----------
        varInDirection : int
            Direction index for self-weight (1 = X, 2 = Y, 3 = Z).
        varLoadFactor : float
            Multiplying factor for self-weight.

        Returns
        -------
        bool
            True if self-weight was added successfully.
            False if an error occurred.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddSelfWeightInXYZ(2, 1.0)
        >>> print(result)
        """
        return self._load.AddSelfWeightInXYZ(varInDirection, varLoadFactor)
```

[docs]

```
    def AddSelfWeightInXYZToGeometry(self, varGeomNumberIDs:list, varInDirectio
        """
        Adds self-weight to specified geometry entities in the active load case

        Parameters
        ----------
        varGeomNumberIDs : list of int
            List of beam, plate, or solid number IDs.
        varInDirection : int
            Direction index for self-weight (1 = X, 2 = Y, 3 = Z).
        varLoadFactor : float
            Multiplying factor for self-weight.

        Returns
        -------
        bool
            True if self-weight was added to the specified geometries.
            False if an error occurred.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddSelfWeightInXYZToGeometry([1,2], 3, 0.9)
        >>> print(result)
        """
        safe_GeomMemNoList = make_safe_array_long_input(varGeomNumberIDs)
        return self._load.AddSelfWeightInXYZToGeometry(safe_GeomMemNoList, varI
```

[docs]

```
    def AddNodalLoad(self, nodeIds:list, forceInXDir: float, forceInYDir: float
        """
        Adds joint load to the specified node numbers.

        Parameters
        ----------
        nodeIds : list of int
            List of node IDs to apply the joint load.
        forceInXDir : float
            Force in the X direction.
        forceInYDir : float
```

```
            Force in the Y direction.
        forceInZDir : float
            Force in the Z direction.
        momentInXDir : float
            Moment in the X direction.
        momentInYDir : float
            Moment in the Y direction.
        momentInZDir : float
            Moment in the Z direction.

        Returns
        -------
        bool
            True if the joint load was added successfully.
            False if an error occurred.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddNodalLoad([1,2], 10, 0, 0, 0, 0, 0)
        >>> print(result)
        """
        safe_NodeIdList = make_safe_array_long_input(nodeIds)
        return self._load.AddNodalLoad(safe_NodeIdList, forceInXDir, forceInYDi
```

[docs]

```
    def AddSupportDisplacement (self, nodeIds:list, varDirection: int, varDispV
        """
        Adds support displacement to one or more nodes.

        Parameters
        ----------
        nodeIds : list of int
            List of node IDs.
        varDirection : int
            Direction index (1 = X, 2 = Y, 3 = Z).
        varDispValue : float
            Displacement value in the specified direction.

        Returns
        -------
        bool
            True if the support displacement was added successfully.
            False if an error occurred.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddSupportDisplacement([1], 1, 5.0)
        >>> print(result)
        """
        safe_NodeIdList = make_safe_array_long_input(nodeIds)
        return self._load.AddSupportDisplacement(safe_NodeIdList, varDirection,
```

[docs]

```
    def AddMemberUniformForce (self, beamIds:list, varDirection:int, varForce:f
        """
        Adds a uniform force to the specified beams.

        Parameters
        ----------
        beamIds : list of int
            List of beam IDs.
        varDirection : int
            Load direction (1 to 9 for LocalX, LocalY, LocalZ, GlobalX, GlobalY
        varForce : float
            Magnitude of the uniform force.
        varD1 : float
            Distance from the start of the member to the start of the load.
        varD2 : float
            Distance from the start of the member to the end of the load.
        varD3 : float
            Perpendicular distance from the member shear center to the local pla

        Returns
        -------
        bool
            True if the uniform force was added successfully.
            False if an error occurred.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddMemberUniformForce([1,2], 1, 5.0, 0, 5,
        >>> print(result)
        """
        safe_BeamIdList = make_safe_array_long_input(beamIds)
        return self._load.AddMemberUniformForce(safe_BeamIdList, varDirection,
```

[docs]

```
    def AddMemberUniformMoment (self, beamIds:list, varDirection:int, varMoment
        """
        Adds a uniform moment to the specified beams.

        Parameters
        ----------
        beamIds : list of int
            List of beam IDs.
        varDirection : int
            Load direction (1 to 9 for LocalX, LocalY, LocalZ, GlobalX, GlobalY
        varMoment : float
            Magnitude of the uniform moment.
        varD1 : float
            Distance from the start of the member to the start of the load.
        varD2 : float
            Distance from the start of the member to the end of the load.
        varD3 : float
            Perpendicular distance from the member shear center to the local pla

        Returns
        -------
        bool
            True if the uniform moment was added successfully.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddMemberUniformMoment([1], 2, 10.0, 0, 5, 0
        >>> print(result)
        """
        safe_BeamIdList = make_safe_array_long_input(beamIds)
        retval = self._load.AddMemberUniformMoment(safe_BeamIdList, varDirectio
        if retval < 0:
            raise_os_error_if_error_code(retval)
        return bool(retval)
```

[docs]

```
    def AddMemberConcForce(self, beamIds:list, varDirection:int, varForce:float,
        """
        Adds a concentrated force to the specified beams.

        Parameters
        ----------
        beamIds : list of int
            List of beam IDs.
        varDirection : int
            Load direction (1 to 6 for LocalX, LocalY, LocalZ, GlobalX, GlobalY,
        varForce : float
            Magnitude of the concentrate force in current units.
        varD1 : float
            Distance from the start of the member to concentrated force.
        varD2 : float
            Perpendicular distance from the member shear center to the local pla

        Returns
        -------
        int
            Returns 0 OK.
            Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddMemberConcForce([1], 1, 20.0, 2.5, 0)
        >>> print(result)
        """
        safe_BeamIdList = make_safe_array_long_input(beamIds)
        return self._load.AddMemberConcForce(safe_BeamIdList, varDirection, var
```

[docs]

```
    def AddMemberConcMoment(self, beamIds:list, varDirection:int, varMoment:flo
        """
        Adds a concentrated moment to the specified beams.

        Parameters
        ----------
        beamIds : list of int
            List of beam IDs.
        varDirection : int
            Load direction (1 to 6 for LocalX, LocalY, LocalZ, GlobalX, GlobalY,
        varMoment : float
            Magnitude of the concentrate moment in current units.
        varD1 : float
            Distance from the start of the member to concentrated moment.
        varD2 : float
            Perpendicular distance from the member shear center to the local pla

        Returns
        -------
        int
            Returns True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddMemberConcMoment([1], 2, 15.0, 3.0, 0)
        >>> print(result)
        """
        safe_BeamIdList = make_safe_array_long_input(beamIds)
        retval = self._load.AddMemberConcMoment(safe_BeamIdList, varDirection, v
        if retval < 0:
            raise_os_error_if_error_code(retval)
        return bool(retval)
```

[docs]

```
    def AddMemberLinearVari (self, memberIds:list[int], varDirection:int, varW1
        """
        Adds LINEARLY VARYING load to beams.

        Parameters
        ----------
        memberIds : list of int
            List of member IDs.
        varDirection : int
            Load direction (1 to 3 for LocalX, LocalY, LocalZ respectively).
        varW1 : float
            Load at the start of the member.
        varW2 : float
            Load at the end of the member.
        varW3 : float
            Load in the middle of the member (for triangular load).

        Returns
        -------
        int
            Returns 0 OK.
            Returns -1 General error.
            Returns -8001 Load direction is invalid.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddMemberLinearVari([1], 2, 2.0, 0.0, 0.0)
        >>> print(result)
        """
        safe_MemberIdList = make_safe_array_long_input(memberIds)
        return self._load.AddMemberLinearVari(safe_MemberIdList, varDirection, 
```

[docs]

```
    def AddMemberTrapezoidal (self, memberIds:list, varDirection:int, varW1:flo
        """
        Adds trapezoidal linearly varying load to beams.

        Parameters
        ----------
        memberIds : list of int
            List of member IDs.
        varDirection : int
            Load direction (1 to 9 for LocalX, LocalY, LocalZ, GlobalX, GlobalY,
        varW1 : float
            Load at the start of the member.
        varW2 : float
            Load at the end of the member.
        varD1 : float
            Distance from the start of the member to loading starting point.
        varD2 : float
            Distance from the start of the member to loading stopping point.

            Notes:
            - If  varD1 and varD2 are not given, the load is assumed to cover th

        Returns
        -------
        int
            Returns 0 OK.
            Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Load.AddMemberTrapezoidal([1], 1, 5.0, 10.0, 0, 5
        >>> print(result)
        """
        safe_MemberIdList = make_safe_array_long_input(memberIds)
        return self._load.AddMemberTrapezoidal(safe_MemberIdList, varDirection,
```

[docs]

```
    def AddMemberAreaLoad (self, beamIds:list, load:float ):
        """
        Adds AREA LOAD to beams.

        Parameters
        ----------
        beamIds : list of int
            List of Beam IDs.
        load : float
            Magnitude of the load value.

        Returns
        -------
        bool
            Returns 0 if OK.
            Returns -1 if General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddMemberAreaLoad([1], 5.0)
        """
        safe_BeamIdList = make_safe_array_long_input(beamIds)
        return self._load.AddMemberAreaLoad(safe_BeamIdList, load)
```

[docs]

```
    def AddMemberFixedEnd (self, beamIds:list, loadStart:float, loadEnd:float )
        """
        Adds FIXED END LOAD to beams.

        Parameters
        ----------
        beamIds : list of int
```

```
            List of Beam IDs.
        loadStart : list of float
            Load at starting point in form of array containing 6 elements corres
        loadEnd : list of float
            Load at end point in form of array containing 6 elements correspondi

        Returns
        -------
        bool
            Returns True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddMemberFixedEnd([1], [1.0, 1.0, 1.0, 0, 0, 0], [1.0
        """
        safe_BeamIdList = make_safe_array_long_input(beamIds)
        safe_loadStart = make_safe_array_double_input(loadStart)
        safe_loadEnd = make_safe_array_double_input(loadEnd)
        retval = self._load.AddMemberFixedEnd(safe_BeamIdList, safe_loadStart, s
        if not bool(retval):
            raise_os_error_if_error_code(retval)
        return bool(retval)
```

[docs]

```
    def AddElementPressure (self, plateIds:list, varDirection:int, varPressure:
        """
        Adds pressure load to plate elements.

        Parameters
        ----------
        plateIds : list of int
            List of plate IDs.
        varDirection : int
            Load direction: (1 to 9 for LocalX, LocalY, LocalZ, GlobalX, GlobalY
        varPressure : float
            Magnitude of the pressure or concentrate load on the element.
        varX1 : float
            Top-Left coordinate X (local).
        varY1 : float
            Top-Left coordinate Y (local).
        varX2 : float
            Bottom-Right coordinate X (local).
        varY2 : float
            Bottom-Right coordinate Y (local).

        Notes:
        - If X1, Y1, X2 and Y2 are 0, the pressure is applied over the full area
        - If X1, Y1, X2 and Y2 are not 0: Pressure applied over the area between
        - If X1 and Y1 are not 0, but X2 and Y2 are 0: Concentrate load applied

        Returns
        -------
        bool
            Returns 0 if OK.
            Returns -1 if General error.
            Returns -8001 if Load direction is invalid.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddElementPressure([1], 3, 5.0, 0.0, 0.0, 1.0, 1.0)
        """
        safe_PlateIdList = make_safe_array_long_input(plateIds)
        return self._load.AddElementPressure(safe_PlateIdList, varDirection, var
```

[docs]

```
    def AddElementHydrostaticPressure (self, plateIds:list, varLoadDirection:in
        """
        Adds Hydrostatic pressure loading to plate elements.

        Parameters
        ----------
        plateIds : list of int
            List of plate IDs.
        varLoadDirection : int
            Load direction: (= 3 to 6 for LocalZ, GlobalX, GlobalY, GlobalZ, res
        varInterpolateDirection : int
            Interpolate along Global Axis(Int or Long), valid direction codes ar
        varMinLoad : float
            Minimum Pressure load
        varMaxLoad : float
            Maximum Pressure load

        Returns
        -------
        bool
            True if successfully.
            False if an error occurred.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddElementHydrostaticPressure([1], 3, 1, 0.0, 10.0)
        """
        safe_PlateIdList = make_safe_array_long_input(plateIds)
        return self._load.AddElementHydrostaticPressure(safe_PlateIdList, varLoa
```

[docs]

```
    def AddTemperatureLoad (self, elementIds:list, varTempAxialElong:float, varT
        """
        Adds TEMPERATURE LOAD to beam or plate elements.

        Parameters
        ----------
        elementIds : list of int
            List of element IDs.
        varTempAxialElong : float
            Change in temperature.
        varTempDiffTopAndBtm : float
            Temperature difference from the top to the bottom of the element (fo
        varTemDiffSide : float
            Temperature difference from side to side of the element (local Z axi

        Returns
        -------
        int
            Returns 0 OK.
            Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddTemperatureLoad([1], 20.0, 30.0, 40.0)
        """
        safe_ElementIdList = make_safe_array_long_input(elementIds)
        return self._load.AddTemperatureLoad(safe_ElementIdList, varTempAxialEl
```

[docs]

```
    def AddStrainLoad (self, elementIds:list, varAxialElong:float ):
        """
        Adds STRAIN LOAD to beam or plate elements.

        Parameters
        ----------
        elementIds : list of int
            List of element IDs.
        varAxialElong : float
            Initial axial elongation (+)/ shrinkage (-) in member due to misfit,

        Returns
        -------
        int
            Returns 0 OK.
            Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddStrainLoad([1], 0.001)
        """
        safe_ElementIdList = make_safe_array_long_input(elementIds)
        return self._load.AddStrainLoad(safe_ElementIdList, varAxialElong)
```

[docs]

```
    def AddLoadAndFactorToCombination (self, loadCombNo:int, loadNo:int, factor
        """
        Adds a primary load case with specified multiplication factor to an exis

        Parameters
        ----------
        loadCombNo : int
            Load Combination Number.
        loadNo : int
            Load Case Reference ID.
        factor : float
            Multiplication factor for the specified primary load case.

        Returns
        -------
        int
            Returns 0 OK.
            Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddLoadAndFactorToCombination(1, 2, 1.0)
        """
        return self._load.AddLoadAndFactorToCombination(loadCombNo, loadNo, fac
```

[docs]

```
    def AddMemberFloorLoad (self, varPressure: float, varYMIN: float, varYMAX:
        """
        Automatically finds enclosed panels in the given boundary (specified usi

        Parameters
        ----------
        varPressure : float
            Magnitude of the pressure or concentrate load on the element.
        varYMIN : float
            Y range from which the load start (in global coordinate).
        varYMAX : float
            Y range at which the load end (in global coordinate).
        varZMIN : float
            Z range from which the load start (in global coordinate).
        varZMAX : float
            Z range at which the load end (in global coordinate).
        varXMIN : float
            X range from which the load start (in global coordinate).
        varXMAX : float
            X range at which the load end (in global coordinate).

        Returns
        -------
        int
            Returns 1 OK.
            Returns 0 General error.
            Returns -8001 Load direction is invalid.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddMemberFloorLoad(5.0, 0.0, 10.0, 0.0, 10.0, 0.0, 10
        """
        return self._load.AddMemberFloorLoad( varPressure, varYMIN, varYMAX, var
```

[docs]

```
    def AddMemberFloorLoadEx (self, rangeType: int, loadDirection:int, pressure
        """
        Automatically finds enclosed panels in the given boundary (specified usi
        Otherwise adds a FLOOR LOAD with pressure (dPressure) in the Global X/Y/

        Parameters
        ----------
        rangeType : int
            Type of the Range :
                +-------+------------+
                | Value | Range Type |
                +=======+============+
                | 0     | X-RANGE    |
                +-------+------------+
                | 1     | Y-RANGE    |
                +-------+------------+
                | 2     | Z-RANGE    |
                +-------+------------+
                | 3     | Group Load |
                +-------+------------+

        loadDirection:int
            Load direction :
                +-------+-----------+
                | Value | Direction |
                +-------+-----------+
                | 0     | Global X  |
                +-------+-----------+
                | 1     | Global Y  |
                +-------+-----------+
                | 2     | Global Z  |
                +-------+-----------+

        pressure: float
```

```
            Magnitude of the pressure or concentrate load on the elemen
        grpOrOneWay: int
            One-Way Load (if it is either "" or "0") or corresponding group name
            Notes:
            - Group name should be of FLOOR group type.
        yMIN: float
            Y range from which the load start(in global coordinate).
        yMAX: float
            Y range at which the load end(in global coordinate).
        zMIN: float
            Z range from which the load start(in global coordinate).
        zMAX: float
            Z range at which the load end(in global coordinate).
        xMIN: float
            X range from which the load start(in global coordinate).
        xMAX: float
            X range at which the load end(in global coordinate).

        Returns
        -------
        int
            Returns 1 OK.
            Returns 0 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddMemberFloorLoadEx(0, 0, 5.0, 0, 0.0, 10.0, 0.0, 10
        """
        return self._load.AddMemberFloorLoadEx( rangeType, loadDirection, pressu
```

[docs]

```
    def AddElementTrapPressureEx (self, PlateIDs: list, LoadDirection: int, Load
        """
        Adds trapezoidal pressure loading to plate elements.

        Parameters
        ----------
        PlateNo : list of int
            List of Plate IDs.
        LoadDirection : int
            Load direction: (= 3 to 6 for LocalZ, GlobalX, GlobalY, GlobalZ, res
        LoadVaryDirection : int
            Load varying direction: (= 1, 2 ,3 for X, Y and JOINT respectively)
        StartPressure : float
            Pressure at loading starting point.(Node1 when JOINT is selected)
        EndPressure : float
            Pressure at loading ending point.(Node2 when JOINT is selected)
        Pressure3 : float
            Pressure at loading point.(applicable only when JOINT is selected)
        Pressure4 : float
            Pressure at loading point.(applicable only when JOINT is selected)
        Returns
        -------
        bool
            True if OK.
            False if any General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddElementTrapPressureEx([1], 3, 1, 100.0, 200.0, 150
        """
        safe_PlateIDList = make_safe_array_long_input(PlateIDs)
        return self._load.AddElementTrapPressureEx( safe_PlateIDList, LoadDirec
```

[docs]

```
def AddWindDefinition (self, varTypeNo: int, varTypeName: str ):
    """
    Adds a Wind Definition named "varTypeName" with number ID varTypeNo.

    Parameters
    ----------
    varTypeNo : int
        Wind Definition Type number ID.
    varTypeName : string
        String name of this new type.

    Returns
    -------
    int
         Returns 0 OK.
         Returns -1 General error.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.Load.AddWindDefinition(1, "Wind Load 1")
    """
    return self._load.AddWindDefinition( varTypeNo, varTypeName)
```

[docs]

```
def AddWindIntensity (self, varTypeNo: int, varIntensity: list, varHeight: 
    """
    Adds to Wind Definitions Wind Intensity by giving Intensity vs. Height.

    Parameters
    ----------
    varTypeNo : int
```

```
            Wind Definition Type number ID.
        varIntensity : list of float
            Intensity values float list
        varHeight : list of float
            Height value float list.

        Returns
        -------
        int
             Returns 0 OK.
             Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddWindIntensity(1,[5.2], [10.0])
        """
        safe_IntensityList = make_safe_array_double_input(varIntensity)
        safe_HeightList = make_safe_array_double_input(varHeight)
        intensity_array_vt = make_variant_vt_ref(safe_IntensityList,  automatio
        height_array_vt = make_variant_vt_ref(safe_HeightList,  automation.VT_A
        return self._load.AddWindIntensity( varTypeNo, intensity_array_vt, heig
```

[docs]

```
    def AddWindExposure (self, varTypeNo: int, varExposureFactor: float, varNod
        """
        Adds Wind Exposures factor to Wind Definitions and assign to nodes.

        Parameters
        ----------
        varTypeNo : int
            Wind Definition Type number ID.
        varExposureFactor : float
            Exposure factor.
        varNodeArray : list of int
            Node number ID list.

        Returns
        -------
        int
             Returns 0 OK.
             Returns -1 General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddWindExposure(1, 1.0, [1,2,3])
        """
        safe_NodeList = make_safe_array_long_input(varNodeArray)
        node_array_vt = make_variant_vt_ref(safe_NodeList,  automation.VT_ARRAY
        return self._load.AddWindExposure( varTypeNo, varExposureFactor, node_a
```

[docs]

```
def AddWindLoad (self, varTypeNo: int, varDirection: int, dFraction: float,
    """
    Adds a wind load.

    Parameters
    ----------
    varTypeNo : int
        Wind Definition Type number ID.
    varDirection : int
        Wind load direction:
            +-------+-----------+
            | Value | Direction |
            +=======+===========+
            | 1     | Global X  |
            +-------+-----------+
            | 3     | Global Z  |
            +-------+-----------+
            | 4     | Global -X |
            +-------+-----------+
            | 6     | Global -Z |
            +-------+-----------+

    dFraction : float
        Factor to be used to multiply the wind loads. Negative signs may be
    varOpenStructure : int
        For Open-type of structure enter 1  , closed-type of structure 0
    dYMIN : float
        Ymin of GLOBAL Y range in which Wind load applied (assume Y axis is
    dYMAX : float
        Ymax of GLOBAL Y range in which Wind load applied (assume Y axis is
    dZMIN : float
        Zmin of GLOBAL Z range in which Wind load applied (assume Y axis is
    dZMAX : float
        Zmax of GLOBAL Z range in which Wind load applied (assume Y axis is
    dXMIN : float
        Xmin of GLOBAL X range in which Wind load applied (assume Y axis is
    dXMAX : float
        Xmax of GLOBAL X range in which Wind load applied (assume Y axis is

    Returns
    -------
    int
         Returns 0 OK.
         Returns -1 General error.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> staad_obj.Load.AddWindLoad(1, 3, 1.0, 1, 0.0, 10.0, 0.0, 10.0, 0.0,
    """
    return self._load.AddWindLoad(varTypeNo, varDirection, dFraction, varOpe
```

[docs]

```
    def AddSeismicDefinition (self, varType: int, varAccidental: int):
        """
        Adds a Seismic Definition with default parameters.

        Parameters
        ----------
        varType : int
            Type of seismic code:
                +-------+------------------------------+
                | Value | Seismic Code                 |
                +=======+==============================+
                | 0     | UBC 1985                     |
                +-------+------------------------------+
                | 1     | UBC 1994                     |
                +-------+------------------------------+
                | 2     | UBC 1997                     |
                +-------+------------------------------+
                | 3     | Indian: IS 1893-1984         |
                +-------+------------------------------+
                | 4     | Indian: IS 1893-2002/2005    |
                +-------+------------------------------+
                | 5     | IBC 2000                     |
                +-------+------------------------------+
                | 6     | IBC 2003                     |
                +-------+------------------------------+
                | 7     | COLOMBIAN: NSR 98            |
                +-------+------------------------------+
                | 8     | JAPANESE (AIJ)               |
                +-------+------------------------------+
                | 9     | ALGERIAN: RPA                |
                +-------+------------------------------+
                | 10    | MEX: CFE-1993                |
                +-------+------------------------------+
                | 11    | MEX: NTC-1987                |
                +-------+------------------------------+
                | 12    | Indian: IS 1893-2016         |
                +-------+------------------------------+
                | 13    | Indian: IS 1893(Part4) 2015  |
                +-------+------------------------------+
                | 14    | IBC 2006                     |
                +-------+------------------------------+
                | 15    | IBC 2012                     |
                +-------+------------------------------+
                | 16    | IBC 2015                     |
                +-------+------------------------------+
                | 17    | IBC 2018                     |
                +-------+------------------------------+
                | 18    | CANADIAN: NRC-2005           |
                +-------+------------------------------+
                | 19    | CANADIAN: NRC-2010           |
                +-------+------------------------------+
                | 20    | CANADIAN: NRC-1995           |
                +-------+------------------------------+
                | 21    | COLOMBIAN: NSR 2010          |
                +-------+------------------------------+
                | 22    | Chinese: GB50011-2001        |
                +-------+------------------------------+
                | 23    | Chinese: GB50011-2010        |
                +-------+------------------------------+
                | 24    | TURKISH                      |
                +-------+------------------------------+

        varAccidental : int
            '1' to consider accidental torsion else '0' ignore

        Returns
        -------
        bool
             True if successful.
             False if unsucessful

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddSeismicDefinition(1, 0)
        """
        return self._load.AddSeismicDefinition(varType, varAccidental)
```

[docs]

```
    def AddSeismicDefSelfWeight (self, varWeightFactor:float):
        """
        Adds self weight to Seismic Definition.

        Parameters
        ----------
        varWeightFactor : float
            Weight Factor to add to self weight

        Returns
        -------
        bool
             True if successful.
             False if unsucessful

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddSeismicDefSelfWeight(1.0)
        """
        return self._load.AddSeismicDefMemberWeight(varWeightFactor)
```

[docs]

```
    def AddSeismicDefMemberWeight (self, varSeismicType:int, loadType:int, weig
        """
        Adds member concentrated/uniform weight to Seismic Definition.

        Parameters
        ----------
        varSeismicType : int
            Type of seismic code:
                +-------+-----------------------------+
                | Value | Seismic Code                |
                +=======+=============================+
                | 0     | AUTO DETECT                 |
                +-------+-----------------------------+
                | 1     | ALGERIAN: RPA               |
                +-------+-----------------------------+
                | 2     | CANADIAN: NRC-1995          |
                +-------+-----------------------------+
                | 3     | CANADIAN: NRC-2005          |
                +-------+-----------------------------+
                | 4     | CANADIAN: NRC-2010          |
                +-------+-----------------------------+
                | 5     | CANADIAN: NRC-2020          |
                +-------+-----------------------------+
                | 6     | Chinese: GB50011-2001       |
                +-------+-----------------------------+
                | 7     | Chinese: GB50011-2010       |
                +-------+-----------------------------+
                | 8     | COLOMBIAN: NSR 95           |
                +-------+-----------------------------+
                | 9     | COLOMBIAN: NSR 2010         |
                +-------+-----------------------------+
                | 10    | IBC 2000                    |
                +-------+-----------------------------+
                | 11    | IBC 2003 ASCE 7-02          |
                +-------+-----------------------------+
                | 12    | IBC 2006/2009 ASCE 7-05     |
                +-------+-----------------------------+
                | 13    | IBC 2012 ASCE 7-10          |
                +-------+-----------------------------+
                | 14    | IBC 2015 ASCE 7-10          |
                +-------+-----------------------------+
                | 15    | IBC 2018 ASCE 7-16          |
                +-------+-----------------------------+
                | 16    | Indian: IS 1893-1984        |
                +-------+-----------------------------+
                | 17    | Indian: IS 1893-2002/2005   |
                +-------+-----------------------------+
                | 18    | Indian: IS 1893-2016        |
                +-------+-----------------------------+
                | 19    | Indian: IS 1893(Part4) 2015 |
                +-------+-----------------------------+
                | 20    | JAPANESE (AIJ)              |
                +-------+-----------------------------+
                | 21    | MEX: CFE-1993               |
                +-------+-----------------------------+
                | 22    | MEX: NTC-1987               |
                +-------+-----------------------------+
                | 23    | TURKISH                     |
                +-------+-----------------------------+
                | 24    | UBC 1985                    |
                +-------+-----------------------------+
                | 25    | UBC 1994                    |
                +-------+-----------------------------+
                | 26    | UBC 1997                    |
                +-------+-----------------------------+

        loadType : int
            1 for uniform loadType and 2 for concentrated loadType
        weight : float
            Uniform weight.
        startDist : float
            Starting distance( = distance from member starting node to weight st
        endDist : float
             Ending distance( =  distance from member starting node to weight en
        memberList : list of int
            List of Member ID to add member concentrated/uniform weight

        Returns
        -------
        bool
             True if successful adds member concentrated/uniform weight to Seism
             False if unsucessful

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddSeismicDefMemberWeight(1, 2, 3.0, 4.0, 5.0, [6, 7,
        """
        safe_MemberIdList = make_safe_array_long_input(memberList)
        return self._load.AddSeismicDefMemberWeight(varSeismicType, loadType, we
```

[docs]

```
    def AddSeismicDefJointWeight (self, weight: float, nodeList:list):
        """
         Adds joint self weight to Seismic Definition.

        Parameters
        ----------
        weight : float
            Weight value.
        nodeList : list of int
            List of Node number IDs

        Returns
        -------
        int
            Returns 0 if OK .
            Returns -1 if General error.
            Returns -100 if Invalid argument.
            Returns -106 if 1 dimensional array of long expected.
            Returns -113 if Integer array/Integer expected.
            Returns -2006 if Invalid Node Number.
            Returns -8034 if Seisemic Code not found.

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddSeismicDefJointWeight(1.0, [1, 2, 3])
        """
        safe_NodeIdList = make_safe_array_long_input(nodeList)
        return self._load.AddSeismicDefJointWeight(weight, safe_NodeIdList)
```

[docs]

```
    def AddSeismicDefElementWeight (self, pressure:float, elementList:list):
        """
        Adds a pressure to Seismic Definition.

        Parameters
        ----------
        pressure : float
            Pressure Value
        elementList : List of int
            'List of element ID list.

        Returns
        -------
        int
             True if successful.
             False if unsucessful

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddSeismicDefElementWeight(1.0, [1, 2, 3])
        """
        return self._load.AddSeismicDefElementWeight(pressure, elementList)
```

[docs]

```
    def AddSeismicDefFloorWeight (self, rangeType: int, loadDirection:int, pres
        """
        Adds a floor weight to Seismic Definition.

        Parameters
        ----------
        rangeType : int
            Type of the Range :
                +-------+------------+
                | Value | Range Type |
                +=======+============+
                | 0     | X-RANGE    |
                +-------+------------+
                | 1     | Y-RANGE    |
                +-------+------------+
                | 2     | Z-RANGE    |
                +-------+------------+
                | 3     | Group Load |
                +-------+------------+

        loadDirection:int
            Load direction :
                +-------+-----------+
                | Value | Direction |
                +-------+-----------+
                | 0     | Global X  |
                +-------+-----------+
                | 1     | Global Y  |
                +-------+-----------+
                | 2     | Global Z  |
                +-------+-----------+

        pressure: float
            Magnitude of the pressure or concentrate load on the elemen
        grpOrOneWay: int
            One-Way Load (if it is either "" or "0") or corresponding group name
            Notes:
            - Group name should be of FLOOR group type.
        yMIN: float
            Y range from which the load start(in global coordinate).
        yMAX: float
            Y range at which the load end(in global coordinate).
        zMIN: float
            Z range from which the load start(in global coordinate).
        zMAX: float
            Z range at which the load end(in global coordinate).
        xMIN: float
            X range from which the load start(in global coordinate).
        xMAX: float
            X range at which the load end(in global coordinate).

        Returns
        -------
        int
             True if successful.
             False if unsucessful

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddSeismicDefFloorWeight(0, 1, 2.0, 3, 4.0, 5.0, 6.0
        """
        return self._load.AddSeismicDefFloorWeight (rangeType, loadDirection, p
```

[docs]

```
    def AddSeismicLoad (self, loadDirection: int, factor: float):
        """
        Adds a Seismic Definition with default parameters.

        Parameters
        ----------
        loadDirection : int
            Load direction: (= 0 to 2 for global X, Y and Z, respectively).
        factor : float
            'Multiplication factor to be used to multiply the seismic load.

        Returns
        -------
        int
             Returns 0 if OK.
             Returns -1 if General error.
             Returns -8001 if Load direction is invalid.

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddSeismicLoad(0, 1.0)
        """
        return self._load.AddSeismicLoad(loadDirection, factor)
```

[docs]

```
    def AddAutoLoadCombinations (self, loadCombCode:str, loadCombCategory:str, l
        """
        Automatically adds load combination based on assigned design code and Ca

        Parameters
        ----------
        loadCombCode : str
            Load Combination Code string name (refer to "Codes.ini")
        loadCombCategory : str
            Load Combination Category string name (refer to corresponding rule i
        loadList : list of int
            Load case reference ID(s), Array of Load case numbers. If the array

        Returns
        -------
        int
            Returns load case reference ID with which automatically load combina
            IfnStartLoadCaseNo is invalid Load Case ID already present Load Case

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddAutoLoadCombinations("AISC 9th Ed","2.3 LRFD Gener
        """
        load_list_vt = make_safe_array_long_input(loadList)
        start_load_case = create_variant_int(0)
        start_load_case_vt = make_variant_vt_ref(start_load_case,  automation.V
        result = self._load.AddAutoLoadCombinations(loadCombCode, loadCombCateg
        if result < 0:
            raise_os_error_if_error_code(result)
        return start_load_case_vt[0]
```

[docs]

```
    def AddRepeatLoad (self, varLoadCaseList:list, varFactorList:list):
        """
        Creates a primary load case using combinations of previously defined pri

        Parameters
        ----------
        varLoadCaseList : list of int
            (Primary) load case reference number ID(s) array.
        varFactorList : list of float
            Multiplication factor array.

        Returns
        -------
        int
            Returns 1 if Load Case is added successfully, 0 otherwise.

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddRepeatLoad([1, 2, 3], [1.0, 2.0, 3.0])
        """
        loadCaseIdList_safe_list = make_safe_array_long_input(varLoadCaseList)
        multiplicationList_safe_list = make_safe_array_double_input(varFactorLis
        return self._load.AddRepeatLoad(loadCaseIdList_safe_list, multiplicatior
```

[docs]

```
    def AddRSLoad (self, varType : int, varFactArray : list, varAccOrDis : int,
        """
        Add Response Spectrum Load

        Parameters
        ----------
        varType : int
            Response Spectrum Load type(1=Srss, 2=Cqc, 3=Absolute, 4=Asce, 5=Ter
        varFactArray :list of float
            Factor List.
        varAccOrDis :int
            1 for Acceleration or 0 for Displacement.
        varScale :float
            Scale
        varDampType :int
            Damp Type(1=DAMP, 2=CDAMP, 3 = MDAMP)
        varDampFact :float
            Damp Factor
        varLinOrLog :int
            Interpolation Type: 1 for Logarithmic or 0 for Linear.
        varMis :int
```

```
            Missing Mass(1 for checked, 0 for unchecked)
        varMisFact :float
            Missing Mass Factor
        varZpa :int
            ZPA(1 for checked, 0 for unchecked)
        varZpaFact :float
            ZPA Factor
        varFf1 :int
            Ff1(1 for checked, 0 for unchecked)
        varFf1Fact :float
            Ff1 Factor
        varFf2 :int
            Ff2(1 for checked, 0 for unchecked)
        varFf2Fact :float
            Ff2 Factor
        varSaveFlag :int
            Save Flag(1 for checked, 0 for unchecked)
        varPairs :int
            Disp Or Acc Set pair count
        varDispOrAccSet :List of float
            Disp or Acc list.
        varVals : List of float
            Value list.

        Returns
        -------
        int
             Returns 1 if is successfully adds response spectrum load.
             Returns 0 if general error.

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddRSLoad(1, [1.0, 2.0, 3.0], 1, 1.0, 1, 1.0, 1, 1,
        """
        safe_FactorList = make_safe_array_double_input(varFactArray)
        safe_DispOrAccelarationList = make_safe_array_double_input(varDispOrAcc
        safe_valueList = make_safe_array_double_input(varVals)
        return self._load.AddRepeatLoad(varType, safe_FactorList, varAccOrDis,
```

[docs]

```
    def AddLoadCasesToEnvelop (self, varEnvNo : int, varLoadCaseList : list):
        """
        Adds a list of primary load case(s) to an existed load envelop.

        Parameters
        ----------
        varEnvNo : int
            Load Envelop reference ID
        varLoadCaseList :list of int
            Load cases reference IDs list.

        Returns
        -------
        int
             Returns 1 if OK.
             Returns 0 if general error.

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddLoadCasesToEnvelop(1, [1, 2, 3])
        """
        safe_LoadCaseList = make_safe_array_long_input(varLoadCaseList)
        return self._load.AddLoadCasesToEnvelop(varEnvNo, safe_LoadCaseList)
```

[docs]

```
    def AddReferenceLoad(self, varRefLoadCaseNoIds: list[int], varFactorList: li
        """
        Adds a reference load item to the currently active load case.

        Parameters
        ----------
        varRefLoadCaseNoIds : list of int
            List of reference load case number IDs from Reference Load Definitio
        varFactorList : list of float
            List of corresponding load factors.

        Returns
        -------
        int
            Reference load case number ID.

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddReferenceLoad([1, 2, 3], [1.0, 2.0, 3.0])
        """
        refLoadCaseNoId_safe_list = make_safe_array_long_input(varRefLoadCaseNoI
        ref_factors_safe_list = make_safe_array_double_input(varFactorList)

        retval = self._load.AddReferenceLoad(refLoadCaseNoId_safe_list, ref_fac
        if retval < 0:
            raise_os_error_if_error_code(retval)
        return retval
```

[docs]

```
    def AddSeismicDefWallArea (self, nTypeNo:int, direction: str, sizeArray:lis
        """
        Adds wall area to Seismic Definition.
        Note:
            - Wall Area is only available in IS1893-2016 seismic code.

        Parameters
        ----------
        nTypeNo : int
            Type of seismic code:
                +-------+-----------------------------------------+
                | Value | Seismic Code                            |
                +=======+=========================================+
                | 15    | Indian: IS 1893-2016                    |
                +-------+-----------------------------------------+
        direction : string
            Direction value. [X directin or Z direction]
        sizeArray : List of float
            Length and Width list consisting of consecutive length and width mea

        Returns
        -------
        int
            Returns 0 if OK .
            Returns -1 if General error.
            Returns -107 if 1 dimensional array of long expected.
            Returns -8034 if Invalid seismic code.
            Returns -8038 if Invalid Direction.

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddSeismicDefWallArea(15, "X", [10.0, 20.0])
        """
        ref_size_safe_list = make_safe_array_double_input(sizeArray)
        refSizeArray_vt = make_variant_vt_ref(ref_size_safe_list,  automation.V
        return self._load.AddSeismicDefWallArea(nTypeNo, direction, refSizeArray
```

[docs]

```
def AddWindDefinitionASCE7Parameters (self, varTypeNo: int, code: int, windS
    """
    Generates the Wind Definition Parameters using ASCE CODE.

    Parameters
    ----------
    varTypeNo : int
        Wind Definition Type number ID
    code : int
        +-------+------------+
        | Value | ASCE CODE  |
        +=======+============+
        | 0     | ACSE 7-1995|
        +-------+------------+
        | 1     | ACSE 7-2002|
        +-------+------------+
        | 2     | ACSE 7-2010|
        +-------+------------+
        | 3     | ACSE 7-2016|
        +-------+------------+

    windSpeed : float
        Wind speed.
    heightAboveSeaLvl : float
```

```
            Ground Height above sea level. [Required only for ASCE7-2016. For ot
        bldgclass : int
            +-------+----------------------------------+
            | value | Building Classification Category |
            +=======+==================================+
            | 0     |Category I                        |
            +-------+----------------------------------+
            | 1     |Category II                       |
            +-------+----------------------------------+
            | 2     |Category III                      |
            +-------+----------------------------------+
            | 3     |Category IV                       |
            +-------+----------------------------------+

        bldgtype : int
            +-------+--------------------------------------+
            | Value | Building Type                        |
            +=======+======================================+
            | 0     | Building Structures                  |
            +-------+--------------------------------------+
            | 1     | Chimney, Tank and similar structures |
            +-------+--------------------------------------+
            | 2     | Solid Signs                          |
            +-------+--------------------------------------+
            | 3     | Open Signs                           |
            +-------+--------------------------------------+
            | 4     | Lattice Framework                    |
            +-------+--------------------------------------+
            | 5     | Trussed Tower                        |
            +-------+--------------------------------------+

        expCat : int
            +-------+-------------------+
            | Value | Exposure Category |
            +=======+===================+
            | 0     | Exposure A        |
            +-------+-------------------+
            | 1     | Exposure B        |
            +-------+-------------------+
            | 2     | Exposure C        |
            +-------+-------------------+
            | 3     | Exposure D        |
            +-------+-------------------+

        varEscarpment : bool
             Consider Wind Speed-up over Hills (FALSE) or Escarpment (TRUE).
        wallType : int
            +-------+-----------+
            | Value | Wall Type |
            +=======+===========+
            | 0     | WindWard  |
            +-------+-----------+
            | 1     | Leeward   |
            +-------+-----------+
            | 2     | SideWall  |
            +-------+-----------+
        varIsFlexible : bool
             Consider structure is Flexible (TRUE) or RIGID (FALSE).

        varEscarpmentData : List of float
            Float list of size 4 containing information describing Hills or Esca
            +-------+----------------------------------------------------------
            | Index | Data
            +=======+==========================================================
            | 0     | Type: 2D Ridge (0)/ 2D Escarpment (1)/ 3D Escarpment (2)
            +-------+----------------------------------------------------------
            | 1     | Height (H)
            +-------+----------------------------------------------------------
            | 2     | Distance upwind of crest (Lh)
            +-------+----------------------------------------------------------
            | 3     | Distance from the crest to the building (x)
            +-------+----------------------------------------------------------

        varbldgData : List of float
            Float list of size 7 containing information describing the building
                - Building Data :
                    +-------+--------------------------------------------------
                    | Index | Item
                    +=======+==================================================
                    | 0     | Enclosure Classification :
                    |       |   - Before 2016 :
                    |       |     Open Building (0)/ Partially Enclosed (1)/ Enc
                    |       |   - [2016] :
                    |       |     Open Building (0)/ Partially Open (1)/ Partial
                    +-------+--------------------------------------------------
                    | 1     | Building Height
                    +-------+--------------------------------------------------
                    | 2     | Building length long the direction of Wind (L)
                    +-------+--------------------------------------------------
                    | 3     | Building length normal to the direction of Wind (
                    +-------+--------------------------------------------------
                    | 4     | Building Natural Frequency
                    +-------+--------------------------------------------------
                    | 5     | Building Damping Ratio
                    +-------+--------------------------------------------------

                - Tank Data :
                    +-------+--------------------------------------------------
                    | Index | Item
                    +=======+==================================================
                    | 0     | Horizontal Cross-section Type :
                    |       |   - Before 2016 :
                    |       |     Square (0)/ Square Diagonal (1)/ Hexagonal or
                    |       |   - [2016] :
                    |       |     Square (0)/ Square Diagonal (1)/ Hexagonal (2)
                    |       |     / Octagonal Axisymmetric (4) / Round Non-axisy
                    +-------+--------------------------------------------------
                    | 1     | Tank Height
                    +-------+--------------------------------------------------
                    | 2     | Least Horizontal Dimension (W)
                    +-------+--------------------------------------------------
                    | 3     | Depth of producing elements like Spoilers and Ribs
                    +-------+--------------------------------------------------
                    | 4     | Structure Natural Frequency
                    +-------+--------------------------------------------------
                    | 5     | Structure Damping Ratio
                    +-------+--------------------------------------------------

                - Solid Sign Data :
                    +-------+--------------------------------------------------
                    | Index | Item
                    +=======+==================================================
                    | 0     | Solid Sign Height (H)
                    +-------+--------------------------------------------------
                    | 1     | Solid Sign M Dimension (M)
                    +-------+--------------------------------------------------
                    | 2     | Solid Sign N Dimension (N)
                    +-------+--------------------------------------------------
                    | 3     | Structure Natural Frequency
                    +-------+--------------------------------------------------
                    | 4     | Structure Damping Ratio
                    +-------+--------------------------------------------------

                - Open Sign/Lattice Framework Data :
                    +-------+--------------------------------------------------
                    | Index | Item
                    +=======+==================================================
                    | 0     | Orientation Type: Flat (0)/ Rounded (1)
                    +-------+--------------------------------------------------
                    | 1     | Height (H)
                    +-------+--------------------------------------------------
                    | 2     | Width
                    +-------+--------------------------------------------------
                    | 3     | Diameter of typical round member
                    +-------+--------------------------------------------------
                    | 4     | Ratio of Solid Area to Gross Area
                    +-------+--------------------------------------------------
                    | 5     | Structure Natural Frequency
                    +-------+--------------------------------------------------
                    | 6     | Structure Damping Ratio
                    +-------+--------------------------------------------------

                - Trussed Tower Data :
                    +-------+--------------------------------------------------
                    | Index | Item
                    +=======+==================================================
                    | 0     | Horizontal Cross Sectio Type: Triangle (0)/ Square
                    +-------+--------------------------------------------------
                    | 1     | Tank Height (H)
                    +-------+--------------------------------------------------
                    | 2     | Width
                    +-------+--------------------------------------------------
                    | 3     | Ratio of Solid Area to Gross Area(in percetage)
                    +-------+--------------------------------------------------
                    | 4     | Structure Natural Frequency
                    +-------+--------------------------------------------------
                    | 5     | Structure Damping Ratio
```

```
                +-------+----------------------------------------------------------

        varUnitsData : List of int
            Float list of size 7 containing Units of data inputs
            +-------+-------------------------------------------------------------
            | Index | Data
            +=======+=============================================================
            | 0     | Unit of Wind Speed {mph(VelocityUnit::mph or 0) or m/sec(V
            +-------+-------------------------------------------------------------
            | 1     | Unit of Height above sea level {inch(LengthUnit::In or 0)
            +-------+-------------------------------------------------------------
            | 2     | [Escarpment] Unit of Height (H) {inch(LengthUnit::In or 0)
            +-------+-------------------------------------------------------------
            | 3     | [Escarpment] Unit of Distance upwind of crest (Lh) {inch(L
            +-------+-------------------------------------------------------------
            | 4     | [Escarpment] Unit of Distance from the crest to the buildi
            +-------+-------------------------------------------------------------
            | 5     | [Building]Unit of Height/ [Tank]Unit of Height/ [Solid Sig
            +-------+-------------------------------------------------------------
            | 6     | [Building]Unit of Length/ [Tank]Unit of Width/ [Solid Sign
            +-------+-------------------------------------------------------------
            | 7     | [Building]Unit of Width/ [Tank]Unit of Depth/ [Solid Sign]
            +-------+-------------------------------------------------------------

        varFactorsUserInput : List of int
            Float list of size 7 containing information describing whether Facto
            +-------+-------------------------------------------------------------
            | Index | Data
            +=======+=============================================================
            | 0     | Kz is User Input(1) or Calculated(0)
            +-------+-------------------------------------------------------------
            | 1     | Kzt is User Input(1) or Calculated(0)
            +-------+-------------------------------------------------------------
            | 2     | I is User Input(1) or Calculated(0)
            +-------+-------------------------------------------------------------
            | 3     | Kd is User Input(1) or Calculated(0)
            +-------+-------------------------------------------------------------
            | 4     | Ke is User Input(1) or Calculated(0) [Required only for AS
            +-------+-------------------------------------------------------------
            | 5     | G is User Input(1) or Calculated(0)
            +-------+-------------------------------------------------------------
            | 6     | Cp is User Input(1) or Calculated(0)
            +-------+-------------------------------------------------------------
            | 7     | Gcpi is User Input(1) or Calculated(0)
            +-------+-------------------------------------------------------------

        varFactors : List of int
            Float list of size 7 containing information describing whether Facto
            +-------+-------------------------------------------+
            | Index | Data                                      |
            +=======+===========================================+
            | 0     | Factor Kz                                 |
            +-------+-------------------------------------------+
            | 1     | Factor Kzt                                |
            +-------+-------------------------------------------+
            | 2     | Factor I                                  |
            +-------+-----------------------------------------+
            | 3     | Factor Kd                               |
            +-------+-----------------------------------------+
            | 4     | Factor Ke [Required only for ASCE7-2016] |
            +-------+-----------------------------------------+
            | 5     | Factor G                                |
            +-------+-----------------------------------------+
            | 6     | Factor Cp                               |
            +-------+-----------------------------------------+
            | 7     | Factor Gcpi                             |
            +-------+-----------------------------------------+

        Returns
        -------
        bool
            Returns True if succesful

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddWindDefinitionASCE7Parameters(1, "ASCE7-10", 100.0
        """
        EscarpmentData = make_safe_array_double_input(varEscarpmentData)

        bldgData_safe_list = make_safe_array_double_input(varbldgData)

        UnitsData_safe_list = make_safe_array_double_input(varUnitsData)

        FactorsUserInput_safe_list = make_safe_array_double_input(varFactorsUser

        Factors_safe_list = make_safe_array_double_input(varFactors)

        retval =  self._load.AddWindDefinitionASCE7Parameters(varTypeNo, code, v
                                                   wallType, varIsFlexib
                                                   FactorsUserInput_safe


        if retval < 0:
            raise_os_error_if_error_code(retval)
        return bool(retval)
```

[docs]

```
    def AddNotionalLoad (self, varPrimaryLoadCaseList: list[int], varPLFactorLis
        """
        Creates a Notional load case using combinations of previously defined pr

        Parameters
        ----------
        varPrimaryLoadCaseList : list of int
            List of Primary load case reference number IDs
        varPLFactorList : list of int
            List of Multiplication factor of Primary load cases
        varPLDirectionList : list of int
            List of Directions of Primary load cases. Directions can be passed a
                +----------------------+----------------------+
                | Direction             | Integer               |
                +=======================+=======================+
                | X OR                  | 1                     |
                | GlobalLoadDirection X | 4                     |
                +-----------------------+-----------------------+
                | Y OR                  | 2                     |
                | GlobalLoadDirection Y | 5                     |
                +-----------------------+-----------------------+
                | Z OR                  | 3                     |
                | GlobalLoadDirection Z | 6                     |
                +-----------------------+-----------------------+

        varReferenceLoadCaseList : list of int
            List of Reference load case reference number IDs
        varRLFactorList : list of int
            List of Multiplication factor of Reference load cases
        varRLDirectionList : list of int
            List of Directions of Reference load cases. Directions can be passed
                +-----------------------+-----------------------+
                | Direction             | Integer               |
                +=======================+=======================+
                | X OR                  | 1                     |
                | GlobalLoadDirection X | 4                     |
                +-----------------------+-----------------------+
                | Y OR                  | 2                     |
                | GlobalLoadDirection Y | 5                     |
                +-----------------------+-----------------------+
                | Z OR                  | 3                     |
                | GlobalLoadDirection Z | 6                     |
                +-----------------------+-----------------------+

        Returns
        -------
        bool
            return True if successful

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddNotionalLoad([1, 2], [1.0, 1.2], [1, 2], [3], [0.8
        """

        ref_PrimaryLoadCase_safe_list = make_safe_array_long_input(varPrimaryLoa
        refPrimaryLoadCaseArray_vt = make_variant_vt_ref(ref_PrimaryLoadCase_sa
        ref_PLFactor_safe_list = make_safe_array_double_input(varPLFactorList)
        refPLFactorArray_vt = make_variant_vt_ref(ref_PLFactor_safe_list,  auto
        ref_PLDirection_safe_list = make_safe_array_long_input(varPLDirectionLis
        refPLDirectionArray_vt = make_variant_vt_ref(ref_PLDirection_safe_list,
        ref_ReferenceLoadCase_safe_list = make_safe_array_long_input(varReferen
        refReferenceLoadCaseArray_vt = make_variant_vt_ref(ref_ReferenceLoadCase
        ref_RLFactor_safe_list = make_safe_array_double_input(varRLFactorList)
        refRLFactorArray_vt = make_variant_vt_ref(ref_RLFactor_safe_list,  auto
        ref_RLDirection_safe_list = make_safe_array_long_input(varRLDirectionLis
        refRLDirectionArray_vt = make_variant_vt_ref(ref_RLDirection_safe_list,

        retval = self._load.AddNotionalLoad(refPrimaryLoadCaseArray_vt, refPLFa
        if retval < 0:
```

```
            raise_os_error_if_error_code(retval)
        return True
```

[docs]

```
    def AddDirectAnalysisDefinitionParameter (self, pParamType: int, members: l
        """
        Adds Direct Analysis Definition (FLEX,AXIAL parameters).

        Parameters
        ----------
        pParamType : int
            Integer indicating type of direct analysis parameter to be added. In
                +--------------+-----------------------------------------+
                | Value        | AnalysisCommand                         |
                +==============+=========================================+
                | FLEX = 0     | DirectAnalysisParameterTypes.FLEX       |
                +--------------+-----------------------------------------+
                | AXIAL = 2    | DirectAnalysisParameterTypes.AXIAL      |
                +--------------+-----------------------------------------+

        members : list of int
            List of Member IDs
        param : float
            FLEX parameter value. [For AXIAL this value is Not Applicable. Shoul

        Returns
        -------
        bool
            Returns TRUE if successful
            Returns FALSE if unsuccessful

        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.AddDirectAnalysisDefinitionParameter(0, [1, 2], 0.5)
        True
        """

        refMembersArray_vt = make_safe_array_long_input(members)

        result =  self._load.AddDirectAnalysisDefinitionParameter(pParamType, re
        return bool(result)
```

[docs]

```
    def AddResponseSpectrumLoadEx (self, rsaCode:int, rsaCombination: int, varSe
        """
        Adds Response Spectrum load item to the currently active load case.

        Parameters
        ----------
        rsaCode : int
    Response Spectrum Loading Code. Refer to the following table for the
rsaCombination : int
    Modal combination rule. (SRSS = 0, ABS = 1, CQC = 2, ASCE = 3, TEN =
varSet1Names : list of string
    List of string containing parameter key words. Refer to the Technica
varSet1Vals : list of float
    List of Parameters values corresponding to the keywords supplied in
varSet2Names : list of string
    List of string containing parameter key words for the spectrum gener
varSet2Vals : list of float
    List of Parameters values corresponding to the keywords supplied in
varDataPairs : list of float
    List of containing pairs of time period and acceleration data. NULL
    Inputs (varSet2Names, varSet2Vals) and (varDataPairs) are mutually e

Notes:
- Techincal Reference sections :
    +---------+---------------------+----------------------------------
    | nTypeNo | Seismic Code        | Parameters
    +=========+=====================+==================================
    | 0       | Generic or Custom   | DEC, ECC, X, Y, Z, ACC, DIS, SCA,
    +---------+---------------------+----------------------------------
    | 1       | IS:1893 Part 1 2002 | TOR, DEC, ECC, X, Y, Z, ACC, DIS,
    |         |                     | SOI, CHE, RF
    +---------+---------------------+----------------------------------
    | 2       | IS:1893 2016        | TOR, DEC, ECC, X, Y, Z, ACC, DIS,
    |         |                     | SOI, CHE, RF
    +---------+---------------------+----------------------------------
    | 4       | ENV 1998-1:1994     | ELA, DES, X, Y, Z, ACC, DAM, CDA,
    |         |                     | SOI, ALP, Q
    +---------+---------------------+----------------------------------
    | 5       | EN 1998-1:2004      | ELA, DES, RS1, RS2, X, Y, Z, ACC,
    |         |                     | SOI, ALP, Q
    +---------+---------------------+----------------------------------
    | 6       | IBC 2006            | X, Y, Z, ACC, DAM, CDA, MDA, LIN,
    |         |                     | ZIP, LAT, LON, SS, S1, SCA, FA, FV
    +---------+---------------------+----------------------------------
    | 7       | IBC 2012            | X, Y, Z, ACC, DAM, CDA, MDA, LIN,
    |         |                     | ZIP, LAT, LON, SS, S1, SCA, FA, FV
    +---------+---------------------+----------------------------------
    | 8       | IBC 2015            | X, Y, Z, ACC, DAM, CDA, MDA, LIN,
    |         |                     | ZIP, LAT, LON, SS, S1, SCA, FA, FV
    +---------+---------------------+----------------------------------
    | 10      | SNiP II-7-81        | A, X, KWX, KX1, Y, KWY, KY1, Z, KW
    +---------+---------------------+----------------------------------
    | 11      | SP 14.13330.2011    | ECC, A, X, Y, Z, ACC, SCA, DAM, LO
    +---------+---------------------+----------------------------------
    | 12      | CANADIAN: NRC-2005  | TOR, DEC, ECC, X, Y, Z, ACC, DIS,
    +---------+---------------------+----------------------------------
    | 13      | CANADIAN: NRC-2010  | TOR, DEC, ECC, X, Y, Z, ACC, DIS,
    +---------+---------------------+----------------------------------
    | 14      | GB 50011 2010       | X, Y, Z, ALP, DAM, CDA, MDA, LIN,
    +---------+---------------------+----------------------------------

- Following values should be specified for INT parameter of GB 50011 201
    +--------------+-------+
    | Fortification | Value |
    | Intensity     |       |
    +===============+=======+
    | 6             | 0     |
    +---------------+-------+
    | 7             | 1     |
    +---------------+-------+
    | 7A            | 2     |
    +---------------+-------+
    | 8             | 3     |
    +---------------+-------+
    | 8A            | 4     |
    +---------------+-------+
    | 9             | 5     |
    +---------------+-------+

Returns
-------
bool
    Returns TRUE if successful
    Returns FALSE if unsuccessful

Examples
--------
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> staad_obj.Load.AddResponseSpectrumLoadEx(0, 0, ['DEC', 'ECC'], [1.0,
"""

Set1Names_safe_list = make_safe_array_string_input(varSet1Names)
Set1Vals_safe_list = make_safe_array_double_input(varSet1Vals)
Set2Names_safe_list = make_safe_array_string_input(varSet2Names)
Set2Vals_safe_list = make_safe_array_double_input(varSet2Vals)
DataPairs_safe_list = make_safe_array_double_input(varDataPairs)

return self._load.AddResponseSpectrumLoadEx(rsaCode, rsaCombination, Se
```

[docs]

```
    def AddAutoCombinationRepeat (self, varCode : str, varCategory : str, varLoa
        """
        Automatically adds repeat load based on assigned design code and Categor

        Parameters
        ----------
        varCode : bool
            Load Combination Code string name (refer to "Codes.ini")
        varCategory : bool
            Load Combination Category string name (refer to corresponding rule i
        varLoadList : List of int
            List of Load case reference IDs. If the array is either null or empt
        varStartLoadCaseNo : int
            (Repeat Load) Load case reference ID with which automatically genera
            If nStartLoadCaseNo is valid, auto repeat load will be created from
            If nStartLoadCaseNo is invalid Load Case ID/already present Load Cas
        varGeneratedLCS : int
            (Repeat Load) The counts of automatically generated repeat loads.
        bVarReference : bool
            Whether include Reference load
        bVarNotional : bool
            Whether include Notional load. If it's True but all Directions are F
        dVarNotionalLoadFactor : float
            If bVarNotional is valid, the value of Notional load factor
        bVarGB50017 : bool
            Consider Notional load factor per GB 50017 Design code
        nVarFloor : int
            The count of floor, it is valid when bVarGB50017 is True only
        bVarX : bool
            Consider X Direction of Notional Load
        bVarNegtiveX : bool
            Consider -X Direction of Notional Load
        bVarZ : bool
            Consider Z Direction of Notional Load
        bVarNegtiveZ : bool
            Consider -Z Direction of Notional Load

        Returns
        -------
        int
            Returns 0 if successful.
            Returns -1 if unsuccessful

        Notes:
            - The default path of Codes.ini under "%localappdata%\Bentley\Engine
        """
        loadList_safe_list = make_safe_array_long_input(varLoadList)
        return self._load.AddAutoCombinationRepeat(varCode, varCategory, loadLis
```

[docs]

```
    def RemoveLoadCasesFromEnvelop(self, varEnvNo : int, varLoadCaseList : list
        """
        Removes a list of primary load case(s) from an existed load envelop.

        Parameters
        ----------
        varEnvNo : int
            Load Envelop reference ID
        varLoadCaseList :list of int
            Load cases reference IDs list.

        Returns
        -------
        int
            Returns 0 if OK
            Returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.RemoveLoadCasesFromEnvelop(1, [2, 3])
        """
        safe_LoadCaseList = make_safe_array_long_input(varLoadCaseList)
        return self._load.RemoveLoadCasesFromEnvelop(varEnvNo, safe_LoadCaseLis
```

[docs]

```
    def RemoveAttribute(self, lLoadCase : int):
        """
        Removes the load attribute specified by lLoadCase.

        Parameters
        ----------
        lLoadCase : int
            Load case reference ID

        Returns
        -------
        int
            Returns 0 if OK
            Returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.RemoveAttribute(1)
        """
        return self._load.RemoveAttribute(lLoadCase)
```

[docs]

```
    def ClearPrimaryLoadCase(self, varLoadCaseNos : list, isReferenceLoad: bool
        """
        Clears the load items in a specified Primary Load cases or Reference Loa

        Parameters
        ----------
        varLoadCaseNos : list
            Primary load case reference ID(s) list.
        isReferenceLoad : bool
            If reference load case(s): True or False.

        Returns
        -------
        int
            Returns 1 if OK
            Returns 0 if failed to delete load(s)
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.ClearPrimaryLoadCase([1, 2, 3], False)
        """
        safe_LoadCaseList = make_safe_array_long_input(varLoadCaseNos)
        return self._load.ClearPrimaryLoadCase(safe_LoadCaseList, isReferenceLoa
```

[docs]

```
    def ClearReferenceLoadCase(self, varLoadCaseNos : list):
        """
        Clears the load items in a specified Primary Load cases or Reference Loa

        Parameters
        ----------
        varLoadCaseNos : list
            Primary load case reference ID(s) list.

        Returns
        -------
        int
            Returns 1 if OK
            Returns 0 if failed to delete load(s)

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.ClearReferenceLoadCase([1, 2, 3])
        """
        safe_LoadCaseList = make_safe_array_long_input(varLoadCaseNos)
        return self._load.ClearReferenceLoadCase(safe_LoadCaseList)
```

[docs]

```
    def IsDynamicLoadIncluded(self, nLoadCase : int):
        """
        Checks if dynamic load included in specified load case.

        Parameters
        ----------
        nLoadCase : int
            Load case reference ID

        Returns
        -------
        int
            Returns 1 if YES
            Returns 0 if NO
            Returns -1 if general error.
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.IsDynamicLoadIncluded(1)
        """
        return self._load.IsDynamicLoadIncluded(nLoadCase)
```

[docs]

```
    def IsCombinationCase(self, nLoadCase : int):
        """
        Checks if specified load case is combination load case.

        Parameters
        ----------
        nLoadCase : int
            Load case reference ID

        Returns
        -------
        int
            Returns 1 if YES
            Returns 0 if NO
            Returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.IsCombinationCase(1)
        """
        return self._load.IsCombinationCase(nLoadCase)
```

[docs]

```
    def SplitLoadsOnBeam(self, varBeamOld : int, varBeamNew : int):
        """
        Split Load from BeamOld to BeamNew.

        Parameters
        ----------
        varBeamOld : int
            Old Beam Id
        varBeamNew : int
            New Beam Id

        Returns
        -------
        int
            Returns 1 (TRUE) if Successful
            Returns 0 (FALSE) if General Error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.SplitLoadsOnBeam(1, 2)
        """
        return self._load.SplitLoadsOnBeam(varBeamOld, varBeamNew)
```

[docs]

```
    def MergeLoadsOnBeam(self, varBeamToKeep : int, varBeamToMerge : int):
        """
        Merge Load from beam to merge.

        Parameters
        ----------
        varBeamToKeep : int
            Beam Id where load to not merge.
        varBeamToMerge : int
            Beam Id to where load to merge.

        Returns
        -------
        bool
            Returns 1 (TRUE) if Successful
            Returns 0 (FALSE) if General Error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.MergeLoadsOnBeam(1, 2)
        """
        return self._load.SplitLoadsOnBeam(varBeamToKeep, varBeamToMerge)
```

[docs]

```
    def BeginLoadMerging(self):
        """
        Begin Load Merging

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.BeginLoadMerging()
        """
        return self._load.BeginLoadMerging()
```

[docs]

```
    def EndLoadMerging(self):
        """
        End Load Merging

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.EndLoadMerging()
        """
        return self._load.EndLoadMerging()
```

[docs]

```
    def ModifySeismicDefinitionParams(self, varParamName: str, varValue: float)
        """
        Modifies or adds a seismic parameter in the existing seismic definition

        Parameters
        ----------
        varParamName : string
            Parameter name for the corresponding code in the seismic definition
        varValue : float
             Value corresponding to the above parameter:
                +-------------------------------+-------------------------------
                | Seismic Code                  | Parameters
                +===============================+===============================
                | ALGERIAN: RPA                 | A Q RX RZ STYPE CT CRDAMP PX P
                +-------------------------------+-------------------------------
                | CANADIAN: NRC-1995            | V ZA ZV RX RZ I F CT PX PZ
                +-------------------------------+-------------------------------
                | CANADIAN: NRC-2005            | SA1 SA2 SA3 SA4 IE SCLASS MVX
                +-------------------------------+-------------------------------
                | CANADIAN: NRC-2010            | SA1 SA2 SA3 SA4 I SCLASS MVX M
                +-------------------------------+-------------------------------
                | CHINESE: GB50011-2001         | INTENSITY FREQUENT RARE GROUP
                |                               | Note: For CHINESE: GB50011-200
                +-------------------------------+-------------------------------
                | CHINESE: GB50011-2010         | INTENSITY FREQUENT FORTIFIED F
                |                               | Note: For CHINESE: GB50011-201
                +-------------------------------+-------------------------------
                | COLOMBIAN: NSR 98             | ZONE I S
                +-------------------------------+-------------------------------
                | COLOMBIAN: NSR 2010           | AA AV FA FV I CT PX PZ ALPHA
                +-------------------------------+------------------------------
                | IBC 2000                      | SDS SD1 S1 I RX RZ SCLASS CT F
                +-------------------------------+------------------------------
                | IBC 2003                      | SDS SD1 S1 I RX RZ SCLASS CT F
                +-------------------------------+------------------------------
                | IBC 2006                      | SS S1 ZIP I RX RZ SCLASS CTX C
                +-------------------------------+------------------------------
                | IBC 2012                      | SS S1 ZIP I RX RZ SCLASS CTX C
                +-------------------------------+------------------------------
                | IBC 2015                      | SS S1 ZIP I RX RZ SCLASS CTX C
                +-------------------------------+------------------------------
                | IBC 2018                      | SS S1 ZIP I RX RZ SCLASS CTX C
                |                               | Note: For IBC 2006 - 2018 Plea
                +-------------------------------+------------------------------
                | INDIAN: IS 1893-1984          | ZONE K I B PX PZ
                +-------------------------------+------------------------------
                | INDIAN: IS 1893-2002/2005     | ZONE RF I SS ST DM PX PZ DT GL
                +-------------------------------+------------------------------
                | INDIAN: IS 1893-2016          | ZONE RF I SS ST DM PX PZ DT GL
                +-------------------------------+------------------------------
                | INDIAN: IS 1893(Part4) 2015   | ZONE RF I SS ST DM PX PZ SA DF
                +-------------------------------+------------------------------
                | JAPANESE (AIJ)                | ZONE CO TC ALPHA
                +-------------------------------+------------------------------
                | MEX: CFE-1993                 | ZONE QX QZ GROUP STYPE REGULAF
                +-------------------------------+------------------------------
                | MEX: NTC-1987                 | ZONE QX QZ GROUP SHADOWED REGU
                |                               | Note: For SHADOWED, REGULAR ar
                +-------------------------------+------------------------------
                | TURKISH                       | A TA TB I RX RZ CT PX PZ
                +-------------------------------+------------------------------
                | UBC 1985                      | ZONE I K TS
                +-------------------------------+------------------------------
                | UBC 1994                      | ZONE I RWX RWZ S CT PX PZ
                +-------------------------------+------------------------------
                | UBC 1997                      | ZONE I RWX RWZ STYPE CT PX PZ
                +-------------------------------+------------------------------

        Returns
        -------
        int
            Returns 0 if OK
            Returns -1 if general error.

        Example
        -------
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.ModifySeismicDefinitionParams('ZONE', 0.2)
        """
        return self._load.ModifySeismicDefinitionParams(varParamName, varValue)
```

[docs]

```
    def ComputeWallWindPressureProfile(self, loadingCode:int, windSpeed: float,
        """
        Generates the wall wind pressure profile using ASCE CODE.

        Parameters
        ----------
        loadingCode : int
            ASCE CODE:
                +------------------+--------------------+
                | Value            | ASCE CODE          |
                +==================+====================+
                | ASCE7Y95 = 0     | ACSE 7-1995        |
                +------------------+--------------------+
                | ACSE702 = 1      | ACSE 7-2002        |
                +------------------+--------------------+
                | ACSE705_10 = 2   | ACSE 7-2010        |
                +------------------+--------------------+

        windSpeed : float
            Wind speed. Default value 85 mph.
        bldgClass : int
            Building Classification Category:
                +--------------+------------------------------------+
                | Value        | Building Classification Category   |
                +==============+====================================+
                | TypeI = 0    | Category I                         |
                +--------------+------------------------------------+
                | TypeII = 1   | Category II                        |
                +--------------+------------------------------------+
                | TypeIII = 2  | Category III                       |
                +--------------+------------------------------------+
                | TypeIV = 3   | Category IV                        |
                +--------------+------------------------------------+

        bldgtype : int
            Structure Type:
                +------------------+--------------------------------------+
                | Value            | Structure Type                       |
                +==================+======================================+
                | Building = 0     | Building Structures                  |
                +------------------+--------------------------------------+
                | Chimney = 1      | Chimney, Tank and similar structures |
                +------------------+--------------------------------------+
                | Solidsign = 2    | Solid Signs                          |
                +------------------+--------------------------------------+
                | Opensign = 3     | Open Signs                           |
                +------------------+--------------------------------------+
                | Laticeframe = 4  | Lattice Framework                    |
                +------------------+--------------------------------------+
                | Trusstower = 5   | Trussed Tower                        |
                +------------------+--------------------------------------+

        expCat : int
```

```
            Exposure Category:
                +------------------+---------------------------------------+
                | Value            | Exposure Category                     |
                +==================+=======================================+
                | ExpA = 0         | Exposure A                            |
                +------------------+---------------------------------------+
                | ExpB = 1         | Exposure B                            |
                +------------------+---------------------------------------+
                | ExpC = 2         | Exposure C                            |
                +------------------+---------------------------------------+
                | ExpD = 3         | Exposure D                            |
                +------------------+---------------------------------------+

        bEscarpment : bool
            Consider Wind Speed-up over Hills (FALSE) or Escarpment (TRUE).
        varUnitsData : list of long
            Integer list of size 8 containing Units of data inputs:
                +------------------+--------------------------------------------
                | Index            | Data
                +==================+============================================
                | 0                | Unit of Wind Speed {mph(VelocityUnit::mph c
                +------------------+--------------------------------------------
                | 1                | Unit of Ground height above sea level {inch
                +------------------+--------------------------------------------
                | 2                | [Escarpment] Unit of Height (H) {inch(Lengt
                +------------------+--------------------------------------------
                | 3                | [Escarpment] Unit of Distance upwind of cre
                +------------------+--------------------------------------------
                | 4                | [Escarpment] Unit of Distance from the cres
                +------------------+--------------------------------------------
                | 5                | [Building]Unit of Height/ [Tank]Unit of Hei
                +------------------+--------------------------------------------
                | 6                | [Building]Unit of Length/ [Tank]Unit of Wid
                +------------------+--------------------------------------------
                | 7                | [Building]Unit of Width/ [Tank]Unit of Dept
                +------------------+--------------------------------------------

        varescarpmentData : list
            Information describing Hills or Escarpment:
                +-------+---------------------------------------------------------
                | Index | Data
                +=======+=========================================================
                | 0     | Type: 2D Ridge (0), 2D Escarpment (1), 3D Escarpment (
                +-------+---------------------------------------------------------
                | 1     | Height (H)
                +-------+---------------------------------------------------------
                | 2     | Distance upwind of crest (Lh)
                +-------+---------------------------------------------------------
                | 3     | Distance from the crest to the building (x)
                +-------+---------------------------------------------------------

        varbldgData : list
            List of size 7 containing information describing the building based
                - Building Data :
                    +-------+-------------------------------------------------
                    | Index | Item
```

```
    +=======+=====================================================
    | 0     | Enclosure Classification: Open Building (0)/ Parti
    +-------+-----------------------------------------------------
    | 1     | Building Height
    +-------+-----------------------------------------------------
    | 2     | Building length long the direction of Wind (L)
    +-------+-----------------------------------------------------
    | 3     | Building length normal to the direction of Wind (
    +-------+-----------------------------------------------------
    | 4     | Building Natural Frequency
    +-------+-----------------------------------------------------
    | 5     | Building Damping Ratio
    +-------+-----------------------------------------------------

- OR Tank Data :
    +-------+-----------------------------------------------------
    | Index | Item
    +=======+=====================================================
    | 0     | Horizontal Cross-section Type:- Square (0)/ Square
    +-------+-----------------------------------------------------
    | 1     | Tank Height (H)
    +-------+-----------------------------------------------------
    | 2     | Least Horizontal Dimension (W)
    +-------+-----------------------------------------------------
    | 3     | Depth of producing elements like Spoilers and Ribs
    +-------+-----------------------------------------------------
    | 4     | Structure Natural Frequency
    +-------+-----------------------------------------------------
    | 5     | Structure Damping Ratio
    +-------+-----------------------------------------------------

- OR Solid Sign Data :
    +-------+-----------------------------+
    | Index | Item                        |
    +=======+=============================+
    | 0     | Solid Sign Height  (H)      |
    +-------+-----------------------------+
    | 1     | Solid Sign M Dimension (M)  |
    +-------+-----------------------------+
    | 2     | Solid Sign N Dimension (N)  |
    +-------+-----------------------------+
    | 3     | Structure Natural Frequency |
    +-------+-----------------------------+
    | 4     | Structure Damping Ratio     |
    +-------+-----------------------------+

- OR Open Sign/Lattice Framework Data :
    +-------+-----------------------------------------+
    | Index | Item                                    |
    +=======+=========================================+
    | 0     | Orientation Type: Flat (0)/ Rounded (1) |
    +-------+-----------------------------------------+
    | 1     | Height (H)                              |
    +-------+-----------------------------------------+
    | 2     | Width                                   |
    +-------+-----------------------------------------+
            | 3     | Diameter of typical round member        |
            +-------+-----------------------------------------+
            | 4     | Structure Natural Frequency             |
            +-------+-----------------------------------------+
            | 5     | Structure Damping Ratio                 |
            +-------+-----------------------------------------+
            | 6     | Ratio of Solid Area to Gross Area       |
            +-------+-----------------------------------------+

        - OR Trussed Tower Data :
            +-------+--------------------------------------------------
            | Index | Item
            +=======+==================================================
            | 0     | Horizontal Cross Sectio Type: Triangle (0)/ Square
            +-------+--------------------------------------------------
            | 1     | Height (H)
            +-------+--------------------------------------------------
            | 2     | Width
            +-------+--------------------------------------------------
            | 3     | Structure Natural Frequency
            +-------+--------------------------------------------------
            | 4     | Structure Damping Ratio
            +-------+--------------------------------------------------
            | 5     | Ratio of Solid Area to Gross Area(in percetage)
            +-------+--------------------------------------------------

wallType : int
    Building wall to generate Wind Load on:
        +------------------+--------------------------------------+
        | Value            | Wall Type                            |
        +==================+======================================+
        | WindWard = 0     | WindWard                             |
        +------------------+--------------------------------------+
        | LeeWard = 1      | Leeward                              |
        +------------------+--------------------------------------+
        | SideWall = 2     | SideWall                             |
        +------------------+--------------------------------------+

        - (0 to 2 for WindWard, Leeward and SideWall, respectively).

Returns
-------
int
    Returns number of Height or Intensity data.

Examples
--------
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> staad_obj.Load.ComputeWallWindPressureProfile(2, 90.0, 1, 0, 2, Fals
"""
safe_UnitsDataList = make_safe_array_long_input(varUnitsData)
safe_EscarpmentDataList = make_safe_array_double_input(varEscarpmentData
safe_BldgDataList = make_safe_array_double_input(varBldgData)
retval = self._load.ComputeWallWindPressureProfile(loadingCode, windSpee
                                                   int(bEscarpment), sa
if retval < 0:
    raise_os_error_if_error_code(retval)
return retval
```

[docs]

```
    def ComputeWallWindPressureProfileASCE72016(self, windSpeed: float, heightAl
        """
        Modifies or adds a seismic parameter in the existing seismic definition

        Parameters
        ----------
        windSpeed : float
            Wind speed. Default value 85 mph.
        heightAboveSeaLvl : float
            Ground height above sea level. Used only for ASCE7-2016 Wind. Defaul
        bldgClass : int
            Building Classification Category:
                +--------------+-------------------------------------+
                | Value        | Building Classification Category    |
                +==============+=====================================+
                | TypeI = 0    | Category I                          |
                +--------------+-------------------------------------+
                | TypeII = 1   | Category II                         |
                +--------------+-------------------------------------+
                | TypeIII = 2  | Category III                        |
                +--------------+-------------------------------------+
                | TypeIV = 3   | Category IV                         |
                +--------------+-------------------------------------+

        bldgtype : int
            Structure Type:
                +------------------+---------------------------------------+
                | Value            | Structure Type                        |
                +==================+=======================================+
                | Building = 0     | Building Structures                   |
                +------------------+---------------------------------------+
                | Chimney = 1      | Chimney, Tank and similar structures |
                +------------------+---------------------------------------+
                | Solidsign = 2    | Solid Signs                           |
                +------------------+---------------------------------------+
                | Opensign = 3     | Open Signs                            |
                +------------------+---------------------------------------+
                | Laticeframe = 4  | Lattice Framework                     |
                +------------------+---------------------------------------+
                | Trusstower = 5   | Trussed Tower                         |
                +------------------+---------------------------------------+

        expCat : int
            Exposure Category:
                +------------------+---------------------------------------+
                | Value            | Exposure Category                     |
                +==================+=======================================+
                | ExpA = 0         | Exposure A                            |
                +-------------------+---------------------------------------+
                | ExpB = 1          | Exposure B                            |
                +-------------------+---------------------------------------+
                | ExpC = 2          | Exposure C                            |
                +-------------------+---------------------------------------+
                | ExpD = 3          | Exposure D                            |
                +-------------------+---------------------------------------+

        bEscarpment : bool
            Consider Wind Speed-up over Hills (FALSE) or Escarpment (TRUE).
        varUnitsData : list of long
            Integer list of size 8 containing Units of data inputs:
                +-------------------+---------------------------------------------
                | Index             | Data
                +===================+=============================================
                | 0                 | Unit of Wind Speed {mph(VelocityUnit::mph (
                +-------------------+---------------------------------------------
                | 1                 | Unit of Ground height above sea level {inch
                +-------------------+---------------------------------------------
                | 2                 | [Escarpment] Unit of Height (H) {inch(Lengt
                +-------------------+---------------------------------------------
                | 3                 | [Escarpment] Unit of Distance upwind of cre
                +-------------------+---------------------------------------------
                | 4                 | [Escarpment] Unit of Distance from the cres
                +-------------------+---------------------------------------------
                | 5                 | [Building]Unit of Height/ [Tank]Unit of Hei
                +-------------------+---------------------------------------------
                | 6                 | [Building]Unit of Length/ [Tank]Unit of Wid
                +-------------------+---------------------------------------------
                | 7                 | [Building]Unit of Width/ [Tank]Unit of Dept
                +-------------------+---------------------------------------------

        varescarpmentData : list
            Information describing Hills or Escarpment:
                +-------+-----------------------------------------------------
                | Index | Data
                +=======+=====================================================
                | 0     | Type: 2D Ridge (0), 2D Escarpment (1), 3D Escarpment (
                +-------+-----------------------------------------------------
                | 1     | Height (H)
                +-------+-----------------------------------------------------
                | 2     | Distance upwind of crest (Lh)
                +-------+-----------------------------------------------------
                | 3     | Distance from the crest to the building (x)
                +-------+-----------------------------------------------------

        varbldgData : list
            List of size 7 containing information describing the building based
                - Building Data :
                    +-------+-------------------------------------------------
                    | Index | Item
                    +=======+=================================================
                    | 0     | Enclosure Classification: Open Building (0)/ Parti
                    +-------+-------------------------------------------------
                    | 1     | Building Height
                    +-------+-------------------------------------------------
                    | 2     | Building length long the direction of Wind (L)
                    +-------+--------------------------------------------------
                    | 3     | Building length normal to the direction of Wind (
                    +-------+--------------------------------------------------
                    | 4     | Building Natural Frequency
                    +-------+--------------------------------------------------
                    | 5     | Building Damping Ratio
                    +-------+--------------------------------------------------

                - OR Tank Data :
                    +-------+--------------------------------------------------
                    | Index | Item
                    +=======+==================================================
                    | 0     | Horizontal Cross-section Type:- Square (0)/ Square
                    +-------+--------------------------------------------------
                    | 1     | Tank Height (H)
                    +-------+--------------------------------------------------
                    | 2     | Least Horizontal Dimension (W)
                    +-------+--------------------------------------------------
                    | 3     | Depth of producing elements like Spoilers and Ribs
                    +-------+--------------------------------------------------
                    | 4     | Structure Natural Frequency
                    +-------+--------------------------------------------------
                    | 5     | Structure Damping Ratio
                    +-------+--------------------------------------------------

                - OR Solid Sign Data :
                    +-------+-----------------------------+
                    | Index | Item                        |
                    +=======+=============================+
                    | 0     | Solid Sign Height  (H)      |
                    +-------+-----------------------------+
                    | 1     | Solid Sign M Dimension (M)  |
                    +-------+-----------------------------+
                    | 2     | Solid Sign N Dimension (N)  |
                    +-------+-----------------------------+
                    | 3     | Structure Natural Frequency |
                    +-------+-----------------------------+
                    | 4     | Structure Damping Ratio     |
                    +-------+-----------------------------+

                - OR Open Sign/Lattice Framework Data :
                    +-------+-----------------------------------------+
                    | Index | Item                                    |
                    +=======+=========================================+
                    | 0     | Orientation Type: Flat (0)/ Rounded (1) |
                    +-------+-----------------------------------------+
                    | 1     | Height (H)                              |
                    +-------+-----------------------------------------+
                    | 2     | Width                                   |
                    +-------+-----------------------------------------+
                    | 3     | Diameter of typical round member        |
                    +-------+-----------------------------------------+
                    | 4     | Structure Natural Frequency             |
                    +-------+-----------------------------------------+
                    | 5     | Structure Damping Ratio                 |
                    +-------+-----------------------------------------+
                    | 6     | Ratio of Solid Area to Gross Area       |
                    +-------+-----------------------------------------+

                - OR Trussed Tower Data :
                    +-------+--------------------------------------------------
                    | Index | Item
                    +=======+==================================================
                    | 0     | Horizontal Cross Sectio Type: Triangle (0)/ Square
                    +-------+--------------------------------------------------
                    | 1     | Height (H)
                    +-------+--------------------------------------------------
                    | 2     | Width
                    +-------+--------------------------------------------------
                    | 3     | Structure Natural Frequency
                    +-------+--------------------------------------------------
                    | 4     | Structure Damping Ratio
                    +-------+--------------------------------------------------
                    | 5     | Ratio of Solid Area to Gross Area(in percetage)
                    +-------+--------------------------------------------------

        wallType : int
            Building wall to generate Wind Load on:
                +------------------+---------------------------------------+
                | Value            | Wall Type                             |
                +==================+=======================================+
                | WindWard = 0     | WindWard                              |
                +------------------+---------------------------------------+
                | LeeWard = 1      | Leeward                               |
                +------------------+---------------------------------------+
                | SideWall = 2     | SideWall                              |
                +------------------+---------------------------------------+

                - (0 to 2 for WindWard, Leeward and SideWall, respectively).

        Returns
        -------
        int
            Returns number of Height or Intensity data.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.ComputeWallWindPressureProfileASCE72016(100, 10, 1, 1
        """
        safe_UnitsDataList = make_safe_array_long_input(varUnitsData)
        safe_EscarpmentDataList = make_safe_array_double_input(varEscarpmentData
        safe_BldgDataList = make_safe_array_double_input(varBldgData)
        retval = self._load.ComputeWallWindPressureProfileASCE72016(windSpeed,
        if retval < 0:
            raise_os_error_if_error_code(retval)
        return retval
```

[docs]

```
    def DeleteLoadEnvelop(self, varEnvNo: int):
        """
        Deletes a specified load envelop.

        Parameters
        ----------
        varEnvNo : int
            Load Envelop reference ID.

        Returns
        -------
        int
            Returns 0 if OK
            Returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.DeleteLoadEnvelop(1)
        """
        return self._load.DeleteLoadEnvelop(varEnvNo)
```

[docs]

```
    def DeleteLoadList(self, varLoadListIndex: int):
        """
        Deletes specified load list.

        Parameters
        ----------
        varLoadListIndex : int
            Load list index.

        Returns
        -------
        int
            Returns 0 if OK
            Returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.DeleteLoadList(1)
        """
        return self._load.DeleteLoadList(varLoadListIndex)
```

[docs]

```
    def DeletePrimaryLoadCases(self, varLoadCaseNos: list, varIsReferenceLoads:
        """
        Deletes specified Primary/Reference Load Cases.

        Parameters
        ----------
        varLoadCaseNos : List of int
            List of Primary/Reference load case reference ID.
        varIsReferenceLoads : bool
            If reference load case(s): TRUE or FALSE

        Returns
        -------
        int
            Returns 0 if OK
            Returns -1 if failed to delete load(s)

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.DeletePrimaryLoadCases([1, 2, 3], False)
        """
        loadCaseNoList = make_safe_array_long_input(varLoadCaseNos)
        return self._load.DeletePrimaryLoadCases(loadCaseNoList, varIsReference
```

[docs]

```
    def DeleteReferenceLoadCases(self, varLoadCaseNos: list):
        """
        Deletes specified Reference Load Cases.

        Parameters
        ----------
        varLoadCaseNos : List of int
            List of Reference load case reference ID.

        Returns
        -------
        int
            Returns 0 if OK
            Returns -1 if failed to delete load(s)

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.DeleteReferenceLoadCases([1, 2, 3])
        """
        loadCaseNoList = make_safe_array_long_input(varLoadCaseNos)
        return self._load.DeleteReferenceLoadCases(loadCaseNoList)
```

[docs]

```
    def DeleteWindDefinition(self, nTypeNo: int):
        """
        Deletes Wind definition. All defintions will be deleted if this input is

        Parameters
        ----------
        nTypeNo : int
            Type of Wind.

        Returns
        -------
        int
            Returns 0 if OK
            Returns -8039 if Invalid load definition.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.DeleteWindDefinition(1)
        """
        return self._load.DeleteWindDefinition(nTypeNo)
```

[docs]

```
    def DeleteDirectAnalysisDefinitionParameter(self, pParamType: int):
        """
        Deletes respective parameters from Direct Analysis Definition based on t

        Parameters
        ----------
        pParamType : int
            Integer indicating type of direct analysis parameter to be added. In
                +----------+------------------------------------+
                | Value    | AnalysisCommand                    |
                +==========+====================================+
                |FLEX = 0  | DirectAnalysisParameterTypes.FLEX  |
                +----------+------------------------------------+
                |AXIAL = 2 | DirectAnalysisParameterTypes.AXIAL |
                +----------+------------------------------------+

        Returns
        -------
        int
            Returns True if successful.
            Returns False if unsuccessful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.DeleteDirectAnalysisDefinitionParameter(0)
        """
        return self._load.DeleteDirectAnalysisDefinitionParameter(pParamType)
```

[docs]

```
    def DeleteDirectAnalysisDefinition(self):
        """
        Deletes whole Direct Analysis Definition.

        Returns
        -------
        int
            Returns True if successful.
            Returns False if unsuccessful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.DeleteDirectAnalysisDefinition()
        """
        return self._load.DeleteDirectAnalysisDefinition()
```

[docs]

```
    def GetPrimaryLoadCaseCount(self):
        """
        Returns the total number of primary load cases in the current structure

        Returns
        -------
        int
            Total number of primary load cases.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetPrimaryLoadCaseCount()
        """
        return self._load.GetPrimaryLoadCaseCount()
```

[docs]

```
    def GetPrimaryLoadCaseNumbers(self):
        """
        Retrieves all primary load case numbers.

        Returns
        -------
        list of int
            List of load case reference number IDs.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetPrimaryLoadCaseNumbers()
        """
        primaryLoadCaseCount = self._load.GetPrimaryLoadCaseCount();
        primaryLoadCaseIdList_safe_list = make_safe_array_long(primaryLoadCaseCo
        primaryLoadCaseIdArray = make_variant_vt_ref(primaryLoadCaseIdList_safe_
        self._load.GetPrimaryLoadCaseNumbers(primaryLoadCaseIdArray)
        return list(primaryLoadCaseIdArray[0])
```

[docs]

```
    def GetLoadCombinationCaseCount(self):
        """
        Returns the total number of load combination cases in the current struct

        Returns
        -------
        int
            Total number of load combination cases.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadCombinationCaseCount()
        """
        return self._load.GetLoadCombinationCaseCount()
```

[docs]

```
    def GetLoadCombinationCaseNumbers(self):
        """
        Retrieves all load combination case numbers.

        Returns
        -------
        list of int
            List of load case reference number IDs.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadCombinationCaseNumbers()
        """
        loadCombinationCaseCount = self._load.GetLoadCombinationCaseCount();
        loadCombinationCaseId_safe_list = make_safe_array_long(loadCombinationCa
        loadCombinationLoadCaseIdArray = make_variant_vt_ref(loadCombinationCase
        self._load.GetLoadCombinationCaseNumbers(loadCombinationLoadCaseIdArray)
        return loadCombinationLoadCaseIdArray[0]
```

[docs]

```
    def GetReferenceLoadCount(self):
        """
        Returns the number of reference load items in the currently active load

        Returns
        -------
        int
            Number of reference load items.
            Returns -1 in case of an error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetReferenceLoadCount()
        """
        return self._load.GetReferenceLoadCount()
```

[docs]

```
    def GetReferenceLoadCaseCount(self):
        """
        Returns the number of reference load case items in the currently active

        Returns
        -------
        int
            Number of reference load case items.
            Returns -1 in case of an error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetReferenceLoadCaseCount()
        """
        return self._load.GetReferenceLoadCaseCount()
```

[docs]

```
    def GetReferenceLoadCaseNumbers(self):
        """
        Retrieves reference load case number IDs from Reference Load Definitions
        Returns
        -------
        list of int
            List of reference load case IDs.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetReferenceLoadCaseNumbers()
        """
        refLoadCaseCount = self._load.GetReferenceLoadCaseCount()
        refLoadCaseIdList_safe_list = make_safe_array_long(refLoadCaseCount)
        refLoadCaseIdArray = make_variant_vt_ref(refLoadCaseIdList_safe_list,
        self._load.GetReferenceLoadCaseNumbers(refLoadCaseIdArray);
        return list(refLoadCaseIdArray[0])
```

[docs]

```
    def GetNoOfSetsInReferenceLoad(self, nIndex:int):
        """
        Returns the number of reference load case-factor sets in a specified ref

        Parameters
        ----------
        nIndex : int
            Index of the reference load case item.

        Returns
        -------
        int
            Number of sets in the reference load item.
            Returns -1 in case of an error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNoOfSetsInReferenceLoad(1)
        """
        return self._load.GetNoOfSetsInReferenceLoad(nIndex)
```

[docs]

```
    def GetReferenceLoadByIndex(self, nIndex:int):
        """
        Retrieves a dictionary of load case numbers and their corresponding fact

        Parameters
        ----------
        nIndex : int
```

```
            Index of the reference load.

        Returns
        -------
        tuple of lists
            tuple of load case number factor lists. [[loadcase1, loadcase2, ...]

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetReferenceLoadByIndex(1)
        """
        refLoadCaseCount = self.GetNoOfSetsInReferenceLoad(nIndex)
        if refLoadCaseCount <= 0:
            return
        refLoad_safe_array = make_safe_array_long(refLoadCaseCount)
        refloadArray_vt = make_variant_vt_ref(refLoad_safe_array,  automation.V
        factor_safe_array = make_safe_array_double(refLoadCaseCount)
        refFactorArray_vt = make_variant_vt_ref(factor_safe_array,  automation.V
        retval = self._load.GetReferenceLoadByIndex(nIndex, refloadArray_vt, re
        if retval <= 0:
            return [], []
        return list(refloadArray_vt[0]), list(refFactorArray_vt[0])
```

[docs]

```
    def GetReferenceLoadType(self, varLoadNo:int):
        """
        Returns the type of a reference load.

        Parameters
        ----------
        varLoadNo : int
            Reference load number.

        Returns
        -------
        int
            Reference load type (0 to 23).
            Returns -1 in case of an error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetReferenceLoadType(1)
        """
        return self._load.GetReferenceLoadType(varLoadNo)
```

[docs]

```
    def GetReferenceLoadCaseTitle(self, varLoadNo:int):
        """
        Returns the title of a reference load case.

        Parameters
        ----------
        varLoadNo : int
            Reference load number.

        Returns
        -------
        str
            Title of the reference load case.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetReferenceLoadCaseTitle(1)
        """
        return self._load.GetReferenceLoadCaseTitle(varLoadNo)
```

[docs]

```
    def GetBeamCountAtFloor(self, varfMinX: float, varfMaxX:float, varfMinY: fl
        """
        Get the beam count at the specific floor.

        Parameters
        ----------
        varfMinX : float
            varfMinX X range start (in global coordinate)
        varfMaxX : float
            varfMaxX X range end (in global coordinate)
        varfMinY : float
            varfMinY Y range start (in global coordinate)
        varfMaxY : float
            varfMaxY Y range end (in global coordinate)
        varfMinZ : float
            varfMinZ Z range start (in global coordinate)
        varfMaxZ : float
            varfMaxZ Z range end (in global coordinate)
        varnDirection : int
            varnDirection Direction(1 for XRange, 2 for YRange, 3 for ZRange).

        Returns
        -------
        int
            The beam count at the specific floor.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetBeamCountAtFloor(0, 10, 0, 10, 0, 10, 1)
        """
        return self._load.GetBeamCountAtFloor(varfMinX, varfMaxX, varfMinY, var
```

[docs]

```
    def GetInfluenceArea(self, varfMinX: float, varfMaxX: float, varfMinY: floa
        """
        Returns a dictionary of beam to influence area at the specific floor.

        Parameters
        ----------
        varfMinX : float
            X range start (in global coordinate).
        varfMaxX : float
            X range end (in global coordinate).
        varfMinY : float
            Y range start (in global coordinate).
        varfMaxY : float
            Y range end (in global coordinate).
        varfMinZ : float
            Z range start (in global coordinate).
        varfMaxZ : float
            Z range end (in global coordinate).
        varnDirection : int
            Direction(1 for XRange, 2 for YRange, 3 for ZRange).

        Returns
        -------
        Dictionary
            Returns dictionary have beam id to influence area data.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetInfluenceArea(0, 10, 0, 10, 0, 10, 1)
        """
        beamCount = self._load.GetBeamCountAtFloor(varfMinX, varfMaxX, varfMinY
        beamIdList_safe_list = make_safe_array_long(beamCount)
        influenceAreaList_safe_list = make_safe_array_double(beamCount)
        beamIdList = make_variant_vt_ref(beamIdList_safe_list,  automation.VT_A
        influenceAreaList = make_variant_vt_ref(influenceAreaList_safe_list,  au
        self._load.GetInfluenceArea(varfMinX, varfMaxX, varfMinY, varfMaxY, var
        beamToAreaInfluence = {}
        for i in range (0, beamCount):
            beamToAreaInfluence[beamIdList[0][i]] = influenceAreaList[0][i]
        return beamToAreaInfluence
```

[docs]

```
    def GetActiveLoad(self):
        """
        Returns the current load case number.

        Returns
        -------
        int
            Returns active load case number ID.
            Else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetActiveLoad()
        """
        return self._load.GetActiveLoad()
```

[docs]

```
    def GetNodalLoadCount(self, nNodeNo: int):
        """
        Returns number of nodal loads present for the specified node.

        Parameters
        ----------
        nNodeNo : int
            Node Id

        Returns
        -------
        int
            Returns the number of node(s).
            Else -1 if general error (perhaps load case not found).

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNodalLoadCount(1)
        """
        return self._load.GetNodalLoadCount(nNodeNo)
```

[docs]

```
    def GetNodalLoads(self, nNodeNo: int):
        """
        Returns tuple of list of forces in X direction, forces in Y direction, f

        Parameters
        ----------
        nNodeNo : int
            Node Id
        Returns
        -------
        Tuple
            Returns a tuple of list of forces in X direction, forces in Y direct

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNodalLoads(1)
        """
        nodeCount = self._load.GetNodalLoadCount(nNodeNo)
        varFX_safe_list = make_safe_array_double(nodeCount)
        varFY_safe_list = make_safe_array_double(nodeCount)
        varFZ_safe_list = make_safe_array_double(nodeCount)
        varMX_safe_list = make_safe_array_double(nodeCount)
        varMY_safe_list = make_safe_array_double(nodeCount)
        varMZ_safe_list = make_safe_array_double(nodeCount)
        varFXList = make_variant_vt_ref(varFX_safe_list,  automation.VT_ARRAY |
        varFYList = make_variant_vt_ref(varFY_safe_list,  automation.VT_ARRAY |
        varFZList = make_variant_vt_ref(varFZ_safe_list,  automation.VT_ARRAY |
        varMXList = make_variant_vt_ref(varMX_safe_list,  automation.VT_ARRAY |
        varMYList = make_variant_vt_ref(varMY_safe_list,  automation.VT_ARRAY |
        varMZList = make_variant_vt_ref(varMZ_safe_list,  automation.VT_ARRAY |
        self._load.GetNodalLoads(nodeCount, varFXList, varFYList, varFZList, var
        return (varFXList[0], varFYList[0], varFZList[0], varMXList[0], varMYLis
```

[docs]

```
    def GetUDLLoadCount(self, nBeamNo:int):
        """
        Returns the number of uniformly distributed load(s) present for the spec

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        int
            Returns the number of uniformly distributed load item(s) applied.
            Returns -1 if general error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetUDLLoadCount(1)
        """
        return self._load.GetUDLLoadCount(nBeamNo)
```

[docs]

```
    def GetUDLLoads(self, nBeamNo:int):
        """
        Gets the uniformly distributed load(s) with all the parameters for the s

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        tuple
            Return a tuple of lists in which each list consist of
            - load directions (1 to 9 for LocalX, LocalY, LocalZ, GlobalX, Globa
            - magnitude of uniform force
            - distance from start of member to the start of load
            - distance from start of member to the end of load
            - perpendicular distance from the member shear center to the local p
            [[dirL1, dirL2,..], [forceL1, forceL2,..], [dst_startL1, dst_startL2

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetUDLLoads(1)
        """
        UDLLoadCount = self.GetUDLLoadCount(nBeamNo)
        if UDLLoadCount <= 0:
            return ([],[],[],[],[])
        varDirection_safe_list = make_safe_array_long(UDLLoadCount)
        varForce_safe_list = make_safe_array_double(UDLLoadCount)
        varD1_safe_list = make_safe_array_double(UDLLoadCount)
        varD2_safe_list = make_safe_array_double(UDLLoadCount)
        varD3_safe_list = make_safe_array_double(UDLLoadCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automati
        varForceList = make_variant_vt_ref(varForce_safe_list,  automation.VT_A
        varD1List = make_variant_vt_ref(varD1_safe_list,  automation.VT_ARRAY |
        varD2List = make_variant_vt_ref(varD2_safe_list,  automation.VT_ARRAY |
        varD3List = make_variant_vt_ref(varD3_safe_list,  automation.VT_ARRAY |

        retval = self._load.GetUDLLoads( nBeamNo, varDirectionList, varForceLis
        if not bool(retval):
            return ([],[],[],[],[])
        return (list(varDirection_safe_list[0]), list(varForce_safe_list[0]), li
```

[docs]

```
    def GetUNIMomentCount(self, nBeamNo:int):
        """
        Returns the count of uniformly distributed (UNI) moment applied to the s
        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        int
            Returns the number of uniformly distributed (UNI) moment item(s) app
            Returns -1 if general error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetUNIMomentCount(1)
        """
        return self._load.GetUNIMomentCount(nBeamNo)
```

[docs]

```
    def GetUNIMoments(self, nBeamNo:int):
        """
        Returns the uniformly distributed (UNI) moments with all the parameters

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        List of tuple
            Return a list of tuple in which tuple consist of load direction, mag

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetUNIMoments(1)
        """

        UNILoadCount = self._load.GetUNIMomentCount(nBeamNo)
        varDirection_safe_list = make_safe_array_long(UNILoadCount)
        varMoment_safe_list = make_safe_array_double(UNILoadCount)
        varD1_safe_list = make_safe_array_double(UNILoadCount)
        varD2_safe_list = make_safe_array_double(UNILoadCount)
        varD3_safe_list = make_safe_array_double(UNILoadCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automat
        varMomentList = make_variant_vt_ref(varMoment_safe_list,  automation.VT_
        varD1List = make_variant_vt_ref(varD1_safe_list,  automation.VT_ARRAY |
        varD2List = make_variant_vt_ref(varD2_safe_list,  automation.VT_ARRAY |
        varD3List = make_variant_vt_ref(varD3_safe_list,  automation.VT_ARRAY |
        self._load.GetUNIMoments( nBeamNo, varDirectionList, varMomentList, var
        UNILoads = []
        for i in range (0, UNILoadCount):
            UNILoads.append((varDirectionList[0][i], varMomentList[0][i], varD1
        return UNILoads
```

[docs]

```
    def GetTrapLoadCount(self, nBeamNo:int):
        """
        Returns number of trapezoidal load(s) present for the specified beam.

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        int
            Returns the number of trapezoidal load item(s) applied.
            Returns -1 if general error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetTrapLoadCount(1)
        """
        return self._load.GetTrapLoadCount(nBeamNo)
```

[docs]

```
    def GetTrapLoads(self, nBeamNo:int):
        """
        Returns the trapezoidal load(s) with all the parameters for the specifie

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        List of tuple
            Return a list of tuple in which tuple consist of load direction, Loa
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetTrapLoads(1)
        """
        TrapezodialLoadCount = self._load.GetTrapLoadCount(nBeamNo)
        varDirection_safe_list = make_safe_array_long(TrapezodialLoadCount)
        varW1_safe_list = make_safe_array_double(TrapezodialLoadCount)
        varW2_safe_list = make_safe_array_double(TrapezodialLoadCount)
        varD1_safe_list = make_safe_array_double(TrapezodialLoadCount)
        varD2_safe_list = make_safe_array_double(TrapezodialLoadCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automat
        varW1List = make_variant_vt_ref(varW1_safe_list,  automation.VT_ARRAY |
        varW2List = make_variant_vt_ref(varW2_safe_list,  automation.VT_ARRAY |
        varD1List = make_variant_vt_ref(varD1_safe_list,  automation.VT_ARRAY |
        varD2List = make_variant_vt_ref(varD2_safe_list,  automation.VT_ARRAY |

        self._load.GetTrapLoads( nBeamNo, varDirectionList, varW1List, varW2Lis
        TrapezodialLoads = []
        for i in range (0, TrapezodialLoadCount):
            TrapezodialLoads.append((varDirectionList[0][i], varW1List[0][i], va
        return TrapezodialLoads
```

[docs]

```
    def GetConcForceCount(self, nBeamNo:int):
        """
        Get number of concentrated force(s) present for the specified beam.

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        int
            Returns the number of concentrated force item(s) applied.
            Returns -1 if general error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetConcForceCount(1)
        """
        return self._load.GetConcForceCount(nBeamNo)
```

[docs]

```
    def GetConcForces(self, nBeamNo:int):
        """
        Returns the concentrated force(s) with all the parameters for the speci

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        List of tuple
            Return a list of tuple in which tuple consist of load direction, Mag

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetConcForces(1)
        """

        ConcForceCount = self._load.GetConcForceCount(nBeamNo)
        varDirection_safe_list = make_safe_array_long(ConcForceCount)
        varForce_safe_list = make_safe_array_double(ConcForceCount)
        varD1_safe_list = make_safe_array_double(ConcForceCount)
        varD2_safe_list = make_safe_array_double(ConcForceCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automat
        varForceList = make_variant_vt_ref(varForce_safe_list,  automation.VT_A
        varD1List = make_variant_vt_ref(varD1_safe_list,  automation.VT_ARRAY |
        varD2List = make_variant_vt_ref(varD2_safe_list,  automation.VT_ARRAY |

        self._load.GetConcForces( nBeamNo, varDirectionList, varForceList, varD
        ConcForces = []
        for i in range (0, ConcForceCount):
            ConcForces.append((varDirectionList[0][i], varForceList[0][i], varD
        return ConcForces
```

[docs]

```
    def GetConcMomentCount(self, nBeamNo:int):
        """
        Gets number of concentrated moment(s) present for the specified beam.

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        int
            Returns the number of concentrated moment item(s) applied.
            Returns -1 if general error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetConcMomentCount(1)
        """
        return self._load.GetConcMomentCount(nBeamNo)
```

[docs]

```
    def GetConcMoments(self, nBeamNo:int):
        """
        Returns the concentrated moments(s) with all the parameters for the spe

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        List of tuple
            Return a list of tuple in which tuple consist of load direction, Mag

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetConcMoments(1)
        """

        ConcForceCount = self._load.GetConcForceCount(nBeamNo)
        varDirection_safe_list = make_safe_array_long(ConcForceCount)
        varMoment_safe_list = make_safe_array_double(ConcForceCount)
        varD1_safe_list = make_safe_array_double(ConcForceCount)
        varD2_safe_list = make_safe_array_double(ConcForceCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automat
        varMomentList = make_variant_vt_ref(varMoment_safe_list,  automation.VT_
        varD1List = make_variant_vt_ref(varD1_safe_list,  automation.VT_ARRAY |
        varD2List = make_variant_vt_ref(varD2_safe_list,  automation.VT_ARRAY |

        self._load.GetConcForces( nBeamNo, varDirectionList, varMomentList, var
        ConcForces = []
        for i in range (0, ConcForceCount):
            ConcForces.append((varDirectionList[0][i], varMomentList[0][i], var
        return ConcForces
```

[docs]

```
    def GetNoOfLoadAndFactorPairsForCombination(self, varLoadCombNo: int ):
        """
        Gets the number of load case(s) applied with multiplication factor in sp

        Parameters
        ----------
        varLoadCombNo : int
```

```
            Combination Load case reference number ID.

        Returns
        -------
        int
            Returns the number of load cases in specified load combination.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNoOfLoadAndFactorPairsForCombination(1)
        """
        return self._load.GetNoOfLoadAndFactorPairsForCombination(varLoadCombNo
```

[docs]

```
    def GetLoadAndFactorForCombination(self, varLoadCombNo: int ):
        """
        Get number of concentrated force(s) present for the specified beam.

        Parameters
        ----------
        varLoadCombNo : int
            Combination Load case reference number ID.

        Returns
        -------
        Tuple
            Returns a Tuple consisting of a list of load case reference number I

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadAndFactorForCombination(1)
        """
        LoadCaseCount = self._load.GetNoOfLoadAndFactorPairsForCombination(varLo
        varLoadCaseID_safe_list = make_safe_array_long(LoadCaseCount)
        varMultiplicationFactor_safe_list = make_safe_array_double(LoadCaseCoun
        varLoadCaseIDList = make_variant_vt_ref(varLoadCaseID_safe_list,  automa
        varMultiplicationFactorList = make_variant_vt_ref(varMultiplicationFacto
        self._load.GetLoadAndFactorForCombination(varLoadCombNo, varLoadCaseIDL
        return (varLoadCaseIDList[0], varMultiplicationFactorList[0])
```

[docs]

```
    def GetLoadCaseTitle(self, varLoadNo:int):
        """
        Returns title of the specified load case as a text string. Input 0 to re

        Parameters
        ----------
        varLoadNo : int
            The load case string title.

        Returns
        -------
        str
            Returns the load case string title.
            Returns "NONE" if load case varLoadNo not found.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadCaseTitle(1)
        """
        return self._load.GetLoadCaseTitle(varLoadNo)
```

[docs]

```
    def GetElementPressureLoadCount(self, varPlateNo:int):
        """
        Gets the number pressure load(s) for the specified plate.

        Parameters
        ----------
        varPlateNo : int
            Plate number ID.

        Returns
        -------
        str
            Returns the number of pressure load(s).
            Returns -1 if General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetElementPressureLoadCount(1)
        """
        return self._load.GetElementPressureLoadCount(varPlateNo)
```

[docs]

```
    def GetElementPressureLoads(self, varPlateNo:int):
        """
        Returns the pressure load(s) with all the parameters for the specified p

        Parameters
        ----------
        varPlateNo : int
```

```
            Plate number ID.

        Returns
        -------
        List of tuple
            Returns a list of tuple in which tuple consist of load direction, Ma

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetElementPressureLoads(1)
        """

        PressureLoadCount = self._load.GetElementPressureLoadCount(varPlateNo)
        varDirection_safe_list = make_safe_array_long(PressureLoadCount)
        varW1_safe_list = make_safe_array_double(PressureLoadCount)
        varX1_safe_list = make_safe_array_double(PressureLoadCount)
        varY1_safe_list = make_safe_array_double(PressureLoadCount)
        varX2_safe_list = make_safe_array_double(PressureLoadCount)
        varY2_safe_list = make_safe_array_double(PressureLoadCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automat
        varW1List = make_variant_vt_ref(varW1_safe_list,  automation.VT_ARRAY |
        varX1List = make_variant_vt_ref(varX1_safe_list,  automation.VT_ARRAY |
        varY1List = make_variant_vt_ref(varY1_safe_list,  automation.VT_ARRAY |
        varX2List = make_variant_vt_ref(varX2_safe_list,  automation.VT_ARRAY |
        varY2List = make_variant_vt_ref(varY2_safe_list,  automation.VT_ARRAY |

        self._load.GetElementPressureLoads( PressureLoadCount, varDirectionList,
        PressureLoads = []
        for i in range (0, PressureLoadCount):
            PressureLoads.append((varDirectionList[0][i], varW1List[0][i], varX
        return PressureLoads
```

[docs]

```
    def GetElementConcLoadCount(self, varPlateNo:int):
        """
        Returns the number of concentrated load for specified plate.

        Parameters
        ----------
        varPlateNo : int
            Plate number ID.

        Returns
        -------
        int
            Returns the number of concentrated load on specified plate.
            Returns -1 if General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetElementConcLoadCount(1)
        """
        return self._load.GetElementConcLoadCount(varPlateNo)
```

[docs]

```
    def GetElementConcLoads(self, varPlateNo:int):
        """
        Returns the concentrated load(s) with all the parameters for the specifi

        Parameters
        ----------
        varPlateNo : int
            Plate number ID.

        Returns
        -------
        List of tuple
            Returns a list of tuple in which tuple consist of load direction, pr

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetElementConcLoads(1)
        """

        ConcentratedLoadCount = self.GetElementPressureLoadCount(varPlateNo)
        varDirection_safe_list = make_safe_array_long(ConcentratedLoadCount)
        varW1_safe_list = make_safe_array_double(ConcentratedLoadCount)
        varX1_safe_list = make_safe_array_double(ConcentratedLoadCount)
        varY1_safe_list = make_safe_array_double(ConcentratedLoadCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automat
        varW1List = make_variant_vt_ref(varW1_safe_list,  automation.VT_ARRAY |
        varX1List = make_variant_vt_ref(varX1_safe_list,  automation.VT_ARRAY |
        varY1List = make_variant_vt_ref(varY1_safe_list,  automation.VT_ARRAY |

        self._load.GetElementConcLoads(ConcentratedLoadCount, varDirectionList,
        ConcentratedLoads = []
        for i in range (0, ConcentratedLoadCount):
            ConcentratedLoads.append((varDirectionList[0][i], varW1List[0][i], v
        return ConcentratedLoads
```

[docs]

```
    def GetLoadType(self, varLoadNo:int):
        """
        Returns primary load case category(s) as an long value.

        Parameters
        ----------
        varLoadNo : int
            Primary load case reference ID. Pass in 0 to get information about

        Returns
        -------
        int
            Returns 0 if Dead.
            Returns 1 if Live.
            Returns 2 if Roof Live.
            Returns 3 if Wind.
            Returns 4 if Seismic-H.
            Returns 5 if Seismic-V.
            Returns 6 if Snow.
            Returns 7 if Fluids.
            Returns 8 if Soil.
            Returns 9 if Rain.
            Returns 10 if Ponding.
            Returns 11 if Dust.
            Returns 12 if Traffic.
            Returns 13 if Temperature.
            Returns 14 if Imperfection.
            Returns 15 if Accidental.
            Returns 16 if Flood.
            Returns 17 if Ice.
            Returns 18 if Wind Ice.
            Returns 19 if Crane Hook.
            Returns 20 if Mass.
            Returns 21 if Gravity.
            Returns 22 if Push.
            Returns 23 if None.
            Returns -1 if General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadType(1)
        """
        return self._load.GetLoadType(varLoadNo)
```

[docs]

```
    def GetLoadListCount (self):
        """
        Gets the number of existing load list(s)

        Returns
        -------
        int
            Returns the number of load list(s).
            Returns -1 if General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadListCount()
        """
        return self._load.GetLoadListCount()
```

[docs]

```
    def GetLoadCountInLoadList (self, varLoadListIndex: int):
        """
        Gets the number of load case(s) in specified load list.

        Parameters
        ----------
        varLoadListIndex : int
            Load list index.

        Returns
        -------
        int
            The number of Load Case(s) in specified Load List.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadCountInLoadList(1)
        """
        return self._load.GetLoadCountInLoadList(varLoadListIndex)
```

[docs]

```
    def GetLoadsInLoadList (self, varLoadListIndex: int):
        """
        Gets the load case(s) in specified load list.

        Parameters
        ----------
        varLoadListIndex : int
            Load list index(Starts from one).

        Returns
        -------
        list of int
            Load  Case reference IDs list.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadsInLoadList(1)
        """
        loadListCount = self.GetLoadCountInLoadList(varLoadListIndex)
        if loadListCount == 0:
            return []
        varLoad_safe_list = make_safe_array_long(loadListCount)
        varLoadList = make_variant_vt_ref(varLoad_safe_list,  automation.VT_ARR
        self._load.GetLoadsInLoadList(varLoadListIndex, varLoadList)
        return varLoadList[0]
```

[docs]

```
    def GetAttribute (self, lLoadCase: int):
        """
        Gets load attribute information of specified load case.

        Parameters
        ----------
        lLoadCase : int
            Load case reference ID.

        Returns
        -------
        int
            Returns 0 if OK.
            Returns -1 if General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetAttribute(1)
        """
        return self._load.GetAttribute(lLoadCase)
```

[docs]

```
    def GetRepeatLoadCount (self):
        """
        Returns the number of repeat load commands in the active load case.

        Returns
        -------
        int
            Returns the number of repeat load commands in the active load case.
            Returns 0 if General Error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetRepeatLoadCount()
        """
        return self._load.GetRepeatLoadCount()
```

[docs]

```
    def GetNoLoadFactorInRepeatLoad (self, nIndex: int):
        """
        Returns the number of load and factor pairs associated with a given repe

        Parameters
        ----------
        nIndex : int
            The index(One based) for repeat load.

        Returns
        -------
        int
            Returns number of load and factor pairs associated with a given repe
            Returns -1 if case of invalid repeat load index.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNoLoadFactorInRepeatLoad(1)
        """
        return self._load.GetNoLoadFactorInRepeatLoad(nIndex)
```

[docs]

```
    def GetRepeatLoadByIndex (self, nIndex: int):
        """
        Returns the dictionary of load case IDs to load factors for a given repe

        Parameters
        ----------
        nIndex : int
            The index(One based) for repeat load.

        Returns
        -------
        dictionary
            Returns a dictionary of load case ID to load factor.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetRepeatLoadByIndex(1)
        """

        loadSizeCount = self._load.GetNoLoadFactorInRepeatLoad(nIndex)
        varLoadCase_safe_list = make_safe_array_long(loadSizeCount)
        varLoadFactor_safe_list = make_safe_array_long(loadSizeCount)
        varLoadCaseList = make_variant_vt_ref(varLoadCase_safe_list,  automatio
        varLoadFactorList = make_variant_vt_ref(varLoadFactor_safe_list,  automa
        self._load.GetRepeatLoadByIndex(nIndex, varLoadCaseList, varLoadFactorLi

        loadCaseToFactor = {}
        for i in range (0, loadSizeCount):
            loadCaseToFactor[varLoadCaseList[0][i]] = varLoadFactorList[0][i]

        return loadCaseToFactor
```

[docs]

```
    def GetLinearVaryingLoadCount (self, nBeamNo: int):
        """
        Returns number of linear varying load(s) present for the specified beam

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        int
            Returns the number of linear varying load item(s) applied.
            Returns -1 if General Error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLinearVaryingLoadCount(1)
        """
        return self._load.GetLinearVaryingLoadCount(nBeamNo)
```

[docs]

```
    def GetLinearVaryingLoads (self, nBeamNo: int):
        """
        Returns parameters for defining linear varying loads for specified beam

        Parameters
        ----------
        nBeamNo : int
            Beam number ID.

        Returns
        -------
        List of tuple
            Returns a list of tuple in which tuple consist of load direction, lo
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLinearVaryingLoads(1)
        """

        LinearVaryingLoadCount = self._load.GetLinearVaryingLoadCount(nBeamNo)
        varDirection_safe_list = make_safe_array_long(LinearVaryingLoadCount)
        varW1_safe_list = make_safe_array_double(LinearVaryingLoadCount)
        varW2_safe_list = make_safe_array_double(LinearVaryingLoadCount)
        varW3_safe_list = make_safe_array_double(LinearVaryingLoadCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automat
        varW1List = make_variant_vt_ref(varW1_safe_list,  automation.VT_ARRAY |
        varW2List = make_variant_vt_ref(varW2_safe_list,  automation.VT_ARRAY |
        varW3List = make_variant_vt_ref(varW3_safe_list,  automation.VT_ARRAY |

        self._load.GetLinearVaryingLoads(LinearVaryingLoadCount, varDirectionLis
        LinearVaryingLoads = []
        for i in range (0, LinearVaryingLoadCount):
            LinearVaryingLoads.append((varDirectionList[0][i], varW1List[0][i],
        return LinearVaryingLoads
```

[docs]

```
    def GetLoadTypeCount (self, loadType: int):
        """
        Gets the number of load(s) with specified Load Type in active Load Case

        Parameters
        ----------
        loadType : int
            +-------+-----------------------------------+---+-------+----------
            | Value | LoadType                          | * | Value | LoadType
            +=======+===================================+===+=======+==========
            | 4000  | SelfWeight                        |   | 3275  | Uniform Fo
            +-------+-----------------------------------+---+-------+----------
            | 3110  | Nodal Load (Node)                 |   | 3280  | Uniform Mo
            +-------+-----------------------------------+---+-------+----------
            | 3120  | Nodal Load (Inclined)             |   | 3285  | Concentrat
            +-------+-----------------------------------+---+-------+----------
            | 3910  | Nodal Load (Support Displacement) |   | 3290  | Concentrat
            +-------+-----------------------------------+---+-------+----------
            | 3210  | Uniform Force                     |   | 3295  | Trapezoida
            +-------+-----------------------------------+---+-------+----------
            | 3220  | Uniform Moment                    |   | 3310  | Pressure o
            +-------+-----------------------------------+---+-------+----------
            | 3230  | Concentrated Force                |   | 3310  | Concentrat
            +-------+-----------------------------------+---+-------+----------
            | 3240  | Concentrated Moment               |   | 3310  | Partial pl
            +-------+-----------------------------------+---+-------+----------
            | 3250  | Linear Varying                    |   | 3320  | Trapezoida
            +-------+-----------------------------------+---+-------+-----------
            | 3260  | Trapezoidal                       |   | 3322  | Solid
            +-------+-----------------------------------+---+-------+-----------
            | 3260  | Hydrostatic                       |   | 3710  | Temperatur
            +-------+-----------------------------------+---+-------+-----------
            | 3620  | Pre/Post Stress                   |   | 3720  | Strain
            +-------+-----------------------------------+---+-------+-----------
            | 3810  | Fixed End                         |   | 3721  | Strain Rat
            +-------+-----------------------------------+---+-------+-----------
            | 3530  | FloorLoadGroup                    |   | 3410  | Area
            +-------+-----------------------------------+---+-------+-----------
            | 3554  | OneWayFloorLoadGroup              |   |       |
            +-------+-----------------------------------+---+-------+-----------

        Returns
        -------
        int
            Returns the number of load(s).
            Returns 0 if loadCaseNo not found.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadTypeCount(1)
        """
        return self._load.GetLoadTypeCount(loadType)
```

[docs]

```
    def GetListSizeForLoadType (self, loadType: int, loadIndex: int):
        """
        Gets number of entities to vwhich specified Load Type and load index.

        Parameters
        ----------
        loadType : int
            +-------+-----------------------------------+---+-----+-------------
            | Value | LoadType                          | * | Value | LoadType
            +=======+===================================+===+=======+===========
            | 4000  | SelfWeight                        |   | 3275  | Uniform Fo
            +-------+-----------------------------------+---+-------+-----------
            | 3110  | Nodal Load (Node)                 |   | 3280  | Uniform Mo
            +-------+-----------------------------------+---+-------+-----------
            | 3120  | Nodal Load (Inclined)             |   | 3285  | Concentrat
            +-------+-----------------------------------+---+-------+-----------
            | 3910  | Nodal Load (Support Displacement) |   | 3290  | Concentrat
            +-------+-----------------------------------+---+-------+-----------
            | 3210  | Uniform Force                     |   | 3295  | Trapezoida
            +-------+-----------------------------------+---+-------+-----------
            | 3220  | Uniform Moment                    |   | 3310  | Pressure
            +-------+-----------------------------------+---+-------+-----------
            | 3230  | Concentrated Force                |   | 3310  | Concentrat
            +-------+-----------------------------------+---+-------+-----------
            | 3240  | Concentrated Moment               |   | 3310  | Partial pl
            +-------+-----------------------------------+---+-------+-----------
            | 3250  | Linear Varying                    |   | 3320  | Trapezoida
            +-------+-----------------------------------+---+-------+-----------
            | 3260  | Trapezoidal                       |   | 3322  | Solid
            +-------+-----------------------------------+---+-------+-----------
            | 3260  | Hydrostatic                       |   | 3710  | Temperatur
            +-------+-----------------------------------+---+-------+-----------
            | 3620  | Pre/Post Stress                   |   | 3720  | Strain
            +-------+-----------------------------------+---+-------+-----------
            | 3810  | Fixed End                         |   | 3721  | Strain Rat
            +-------+-----------------------------------+---+-------+-----------
            | 3530  | FloorLoadGroup                    |   | 3410  | Area
            +-------+-----------------------------------+---+-------+-----------
            | 3554  | OneWayFloorLoadGroup              |   |       |
            +-------+-----------------------------------+---+-------+-----------

        loadIndex : int
            Load item index of specified load type (Zero based). Program returns

        Returns
        -------
        int
            Returns the number of entities.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetListSizeForLoadType(1, 0)
        """
        return self._load.GetListSizeForLoadType(loadType, loadIndex)
```

[docs]

```
def GetAssignmentListForLoadType (self, loadType: int, loadIndex: int):
    """
    Return the list of entities that have been assigned to a load command in

    Parameters
    ----------
    loadType : int
        +-------+-----------------------------------+-+-------+-------------
        | Value | LoadType                          | * | Value | LoadType
        +=======+===================================+===+=======+===========
        | 4000  | SelfWeight                        |   | 3275  | Uniform Fo
        +-------+-----------------------------------+---+-------+-----------
        | 3110  | Nodal Load (Node)                 |   | 3280  | Uniform Mo
        +-------+-----------------------------------+---+-------+-----------
        | 3120  | Nodal Load (Inclined)             |   | 3285  | Concentrat
        +-------+-----------------------------------+---+-------+-----------
        | 3910  | Nodal Load (Support Displacement) |   | 3290  | Concentrat
        +-------+-----------------------------------+---+-------+-----------
        | 3210  | Uniform Force                     |   | 3295  | Trapezoida
```

```
            +-------+----------------------------------+---+-------+----------
            | 3220  | Uniform Moment                   |   | 3310  | Pressure
            +-------+----------------------------------+---+-------+----------
            | 3230  | Concentrated Force               |   | 3310  | Concentrat
            +-------+----------------------------------+---+-------+----------
            | 3240  | Concentrated Moment              |   | 3310  | Partial pl
            +-------+----------------------------------+---+-------+----------
            | 3250  | Linear Varying                   |   | 3320  | Trapezoida
            +-------+----------------------------------+---+-------+----------
            | 3260  | Trapezoidal                      |   | 3322  | Solid
            +-------+----------------------------------+---+-------+----------
            | 3260  | Hydrostatic                      |   | 3710  | Temperatur
            +-------+----------------------------------+---+-------+----------
            | 3620  | Pre/Post Stress                  |   | 3720  | Strain
            +-------+----------------------------------+---+-------+----------
            | 3810  | Fixed End                        |   | 3721  | Strain Rat
            +-------+----------------------------------+---+-------+----------
            | 3530  | FloorLoadGroup                   |   | 3410  | Area
            +-------+----------------------------------+---+-------+----------
            | 3554  | OneWayFloorLoadGroup             |   |       |
            +-------+----------------------------------+---+-------+----------

        loadIndex : int
            Load item index of specified load type (Zero based). Program returns

        Returns
        -------
        list of int
            Returns List of Entities number ID(s)

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetAssignmentListForLoadType(1, 0)
        """
        size = self.GetListSizeForLoadType(loadType,loadIndex)
        if size < 1:
            return []
        entities  = make_safe_array_long(size)
        entities_ref = make_variant_vt_ref(entities, automation.VT_ARRAY | auto
        retval = self._load.GetAssignmentListForLoadType(loadType, loadIndex, e
        if retval == 0:
            return []
        return list(entities[0])
```

[docs]

```
    def GetNodalLoadInfo (self, loadIndex: int):
        """
        Gets nodal load(s) generated by specified load item in specified load c

        Parameters
        ----------
        loadIndex : int
            Load item index.

        Returns
        -------
        bool
            Returns a list of 5 nodal forces - FX, FY, FZ, MX, MY and MZ which a

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNodalLoadInfo(1)
        """
        nodalForce_safe_list = make_safe_array_double(6)
        nodalForceList = make_variant_vt_ref(nodalForce_safe_list,  automation.V
        self._load.GetNodalLoadInfo(loadIndex, nodalForceList)
        return nodalForceList[0]
```

[docs]

```
    def GetMemberLoadInfo (self, loadIndex: int):
        """
        Gets member load(s) information generated by specified load item in spec

        Parameters
        ----------
        loadIndex : int
            Load item index (Zero based)

        Returns
        -------
        tuple containing
            - direction: int  (Load direction will be represented numerically - 
            - member force parameters: List [dW1, dW2, dW3]
            - member force distances: List [dD1, dD2, dD3]

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetMemberLoadInfo(1)
        """
        loadCount = 3
        varDirection_safe_list = make_safe_array_long(1)
        varForce_safe_list = make_safe_array_double(loadCount)
        varDistance_safe_list = make_safe_array_double(loadCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list, automatic
        varForceList = make_variant_vt_ref(varForce_safe_list,  automation.VT_AI
        varDistanceList = make_variant_vt_ref(varDistance_safe_list,  automation

        retval = self._load.GetMemberLoadInfo(loadIndex, varDirectionList, varFo
        if not bool(retval):
            return 0, [0,0,0],[0,0,0]
        return varDirectionList[0], varForceList[0], varDistanceList[0]
```

[docs]

```
    def GetElementLoadInfo (self, loadIndex: int):
        """
        Gets element load information generated by specified load item in speci

        Parameters
        ----------
        loadIndex : int
            Load item index (Zero based)

        Returns
        -------
        List of tuple
            Returns a list of tuple in which tuple consist of load direction, e

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetElementLoadInfo(1)
        """
        loadCount = 4
        varDirection_safe_list = make_safe_array_long(loadCount)
        varForce_safe_list = make_safe_array_double(loadCount)
        varDistance_safe_list = make_safe_array_double(loadCount)
        varDirectionList = make_variant_vt_ref(varDirection_safe_list,  automat
        varForceList = make_variant_vt_ref(varForce_safe_list,  automation.VT_A
        varDistanceList = make_variant_vt_ref(varDistance_safe_list,  automatio

        self._load.GetElementLoadInfo(loadCount, varDirectionList, varForceList
        Loads = []
        for i in range (0, loadCount):
            Loads.append((varDirectionList[0][i], varForceList[0][i], varDistan
        return Loads
```

[docs]

```
    def GetNotionalLoadCount (self):
        """
        Returns the number of Notional load.

        Returns
        -------
        int
            Returns the number of Notional load
            Returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNotionalLoadCount()
        """
        return self._load.GetNotionalLoadCount()
```

[docs]

```
    def GetNoLoadFactorDirectionInNotionalLoad (self, nIndex: int):
        """
        Gets the no of factor for specified Notional load.

        Parameters
        ----------
        nIndex : int
            The index for Notional load.

        Returns
        -------
        int
            Returns the factor for specified Notional load.
            Returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNoLoadFactorDirectionInNotionalLoad(1)
        """
        return self._load.GetNoLoadFactorDirectionInNotionalLoad(nIndex)
```

[docs]

```
    def GetNotionalLoadByIndex (self, nIndex: int):
        """
        Gets load case(s), direction(s) and factor(s) for specified Notional loa

        Parameters
        ----------
        nIndex : int
            The index for Notional load.

        Returns
        -------
        List of tuple
            Returns a list of tuple in which tuple consist of load direction, lo

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetNotionalLoadByIndex(1)
        """
        notionalloadCount = self._load.GetNoLoadFactorDirectionInNotionalLoad(n
        Direction_safe_list = make_safe_array_long(notionalloadCount)
        LoadCase_safe_list = make_safe_array_double(notionalloadCount)
        Factor_safe_list = make_safe_array_double(notionalloadCount)
        DirectionList = make_variant_vt_ref(Direction_safe_list,  automation.VT_
        LoadCaseList = make_variant_vt_ref(LoadCase_safe_list,  automation.VT_A
        FactorList = make_variant_vt_ref(Factor_safe_list,  automation.VT_ARRAY

        self._load.GetElementLoadInfo(notionalloadCount, LoadCaseList, FactorLi
        Loads = []
        for i in range (0, notionalloadCount):
            Loads.append((DirectionList[0][i], LoadCaseList[0][i], FactorList[0
        return Loads
```

[docs]

```
    def GetLoadItemsCount (self, loadCaseNo: int):
        """
        Returns the number of loaditems in the specified load case.

        Parameters
        ----------
        loadCaseNo : int
            Load case number.

        Returns
        -------
        int
            Returns the number of loaditems in the specified load case.
            Returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadItemsCount(1)
        """
        return self._load.GetLoadItemsCount(loadCaseNo)
```

[docs]

```
    def GetLoadItemType (self, loadCaseNo: int, loadItemIndex: int):
        """
        Returns the load item type for the specified loadIndex and loadCase.

        Parameters
        ----------
        loadCaseNo : int
            Load case number.
        loadItemIndex : int
```

```
            Load item index (Zero based).

        Returns
        -------
        int
            Returns LoadItemType for the specified loadIndex and loadCase.
            Returns 0 if LoadCase/LoadItemIndex Not Found.
            Return Values and LoadItem Type
            +-------+-----------------------------------+---+-------+-----------
            | Value | LoadItem Type                     | * | Value | LoadItem T
            +=======+===================================+===+=======+===========
            | 4000  | SelfWeight                        |   | 3520  | FloorLoadZ
            +-------+-----------------------------------+---+-------+-----------
            | 3110  | Nodal Load (Node)                 |   | 3530  | FloorLoadC
            +-------+-----------------------------------+---+-------+-----------
            | 3120  | Nodal Load (Inclined)             |   | 3551  | OneWayFloo
            +-------+-----------------------------------+---+-------+-----------
            | 3910  | Nodal Load (Support Displacement) |   | 3552  | OneWayFloo
            +-------+-----------------------------------+---+-------+-----------
            | 3312  | Nodal Load (Region node load)     |   | 3553  | OneWayFloo
            +-------+-----------------------------------+---+-------+-----------
            | 3210  | Uniform Force                     |   | 3554  | OneWayFloo
            +-------+-----------------------------------+---+-------+-----------
            | 3220  | Uniform Moment                    |   | 3310  | Pressure o
            +-------+-----------------------------------+---+-------+-----------
            | 3230  | Concentrated Force                |   | 3311  | Concentrat
            +-------+-----------------------------------+---+-------+-----------
            | 3240  | Concentrated Moment               |   | 3312  | Partial pl
            +-------+-----------------------------------+---+-------+-----------
            | 3250  | Linear Varying                    |   | 3320  | Trapezoida
            +-------+-----------------------------------+---+-------+-----------
            | 3260  | Trapezoidal                       |   | 3322  | Solid
            +-------+-----------------------------------+---+-------+-----------
            | 3261  | Hydrostatic                       |   | 3710  | Temperatur
            +-------+-----------------------------------+---+-------+-----------
            | 3620  | Pre/Post Stress                   |   | 3720  | Strain
            +-------+-----------------------------------+---+-------+-----------
            | 3810  | Fixed End                         |   | 3721  | Strain Rat
            +-------+-----------------------------------+---+-------+-----------
            | 3275  | Uniform Force (Physical)          |   | 4400  | UBC Load
            +-------+-----------------------------------+---+-------+-----------
            | 3280  | Uniform Moment (Physical)         |   | 4600  | Wind Load
            +-------+-----------------------------------+---+-------+-----------
            | 3285  | Concentrated Force (Physical)     |   | 4610  | Wind Load
            +-------+-----------------------------------+---+-------+-----------
            | 3290  | Concentrated Moment (Physical)    |   | 4405  | IbcLoad
            +-------+-----------------------------------+---+-------+-----------
            | 3295  | Trapezoidal (Physical)            |   | 4410  | 1893Load
            +-------+-----------------------------------+---+-------+-----------
            | 3410  | Area                              |   | 4500  | AijLoad
            +-------+-----------------------------------+---+-------+-----------
            | 3510  | FloorLoadYrange                   |   | 4510  | ColombianL
            +-------+-----------------------------------+---+-------+-----------
            | 3511  | FloorLoadXrange                   |   | 4520  | CFELoad
            +-------+-----------------------------------+---+-------+-----------
            | 4570  | TurkishLoad                       |   | 4530  | RPALoad
            +-------+-----------------------------------+---+-------+----------
            | 4575  | GB50011Load                       |   | 4540  | NTCLoad
            +-------+-----------------------------------+---+-------+----------
            | 4576  | Colombian2010Load                 |   | 4550  | NRCLoad
            +-------+-----------------------------------+---+-------+----------
            | 4820  | TimeHistoryLoad                   |   | 4560  | NRCLoad200
            +-------+-----------------------------------+---+-------+----------
            | 4651  | Snow Load Data                    |   | 4561  | NRCLoad201
            +-------+-----------------------------------+---+-------+----------
            | 4201  | Repeat load data                  |   | 4100  | Spectrum L
            +-------+-----------------------------------+---+-------+----------
            | 4223  | Notional Load Data                |   | 4700  | Calulate N
            +-------+-----------------------------------+---+-------+----------
            | 4220  | Reference Load                    |   | 4710  | Modal Calc
            +-------+-----------------------------------+---+-------+----------
            | 4101  | Spectrum Data                     |   | 4222  | Notional L
            +-------+-----------------------------------+---+-------+----------
            | 4701  | Calulate Rayleigh Frequency       |   | 4650  | Snow Load
            +-------+-----------------------------------+---+-------+----------
            | 4200  | Repeat load                       |   |       |
            +-------+-----------------------------------+---+-------+----------

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadItemType(1, 1)

        """
        return self._load.GetLoadItemType(loadCaseNo, loadItemIndex)
```

[docs]

```
    def GetEnvelopeCount (self) :
        """
        Returns number of Envelopes defined.

        Returns
        -------
        int
            Total Number of load Envelopes present.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetEnvelopeCount()
        """
        return self._load.GetEnvelopeCount()
```

[docs]

```
    def GetLoadEnvelopeDetails (self, EnvNo: int):
        """
        Returns

        Parameters
        ----------
        EnvNo : int
            Load Envelope reference ID.

        Returns
        -------
        Tuple
            Returns a tuple containing EnvelopeType and NumberofLoadCasesInEnvel
            Type of Load Envelope
            +-------+-------------------+
            | Value | Load Envelop Type |
            +=======+===================+
            | 0     | NONE              |
            +-------+-------------------+
            | 1     | STRESS            |
            +-------+-------------------+
            | 2     | SERVICEABILITY    |
            +-------+-------------------+
            | 3     | COLUMN            |
            +-------+-------------------+
            | 4     | CONNECTION        |
            +-------+-------------------+
            | 5     | STRENGTH          |
            +-------+-------------------+
            | 6     | TEMPORARY         |
            +-------+-------------------+

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetEnvelopeDetails(1)
        """
        safe_EnvelopeType = make_safe_array_long(0)
        EnvelopeType = make_variant_vt_ref(safe_EnvelopeType,  automation.VT_I4

        safe_NumberofLoadCasesInEnvelope = make_safe_array_long(0)
        NumberofLoadCasesInEnvelope = make_variant_vt_ref(safe_NumberofLoadCases

        self._load.GetLoadEnvelopeDetails(EnvNo, EnvelopeType, NumberofLoadCases
        return (EnvelopeType[0], NumberofLoadCasesInEnvelope[0])
```

[docs]

```
    def GetLoadListfromLoadEnvelope (self, EnvNo: int):
        """
        Gets the list of primary load case reference Ids present in the load env

        Parameters
        ----------
        EnvNo : int
            Load Envelope reference ID

        Returns
        -------
        List of int
            (Primary) load case(s) reference ID(s).

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetLoadListfromLoadEnvelope(1)
        """
        LoadEnvelopeDetails = self.GetLoadEnvelopeDetails(EnvNo)
        LoadCase_safe_list = make_safe_array_long(LoadEnvelopeDetails[1])
        LoadCaseList = make_variant_vt_ref(LoadCase_safe_list,  automation.VT_A

        retval = self._load.GetLoadListfromLoadEnvelope(EnvNo, LoadCaseList)
        if retval <= 0:
            return []
        return list(retval)
```

[docs]

```
    def GetEnvelopeIDs (self):
        """
        Gets the list of Loads Envelope IDs present in the staad file.

        Returns
        -------
        List of int
            Envelope ID(s)

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Load.GetEnvelopeIDs()
        """
        EnvelopeIDCount = self._load.GetEnvelopeCount()
        EnvelopeId_safe_list = make_safe_array_long(EnvelopeIDCount)
        EnvelopeIdList = make_variant_vt_ref(EnvelopeId_safe_list,  automation.
        return self._load.GetEnvelopeIDs(EnvelopeIdList)
```