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

[docs]

```python
class OSDesign:
    CoInitialize()
```

[docs]

```python
    def __init__(self, staadObj):
        self._staad = staadObj
        self._design = self._staad.Design

        self._functions= [
            "CreateDesignBrief",
            "AssignDesignParameter",
            "AssignDesignCommand",
            "AssignDesignGroup",
            "GetDesignBriefCode"
        ]

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



    ## Design FUNCTIONS
```

[docs]

```python
    def CreateDesignBrief(self, design_code: int):
        """
        Create a new design brief with the specified design code.

        Parameters
        ----------
        design_code : int
            Design code index.

        Returns
        -------
        int
            Reference ID of the created design brief.

        Example
        -------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ref_id = staad_obj.Design.CreateDesignBrief(1001)
        """
        return self._design.CreateDesignBrief(design_code)
```

[docs]

```
    def AssignDesignParameter(self, design_ref_id: int, design_param: str, desig
        """
        Assign design parameters to a specified design brief.

        Parameters
        ----------
        design_ref_id : int
            Design brief reference ID.
        design_param : str
            Name of the design parameter.
        design_param_value : str
            Value for the design parameter.
        member_ids : list of int or int
            List of member numbers.

        Returns
        -------
        int
            0 if successful, -1 otherwise.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ref_id = staad_obj.Design.CreateDesignBrief(1001)
        >>> result = staad_obj.Design.AssignDesignParameter(ref_id, "BEAM", "1",
        """
        if isinstance(member_ids, int):
            member_ids = [member_ids]
        safe_members = make_safe_array_long_input(member_ids)
        members_variant = make_variant_vt_ref(safe_members, automation.VT_ARRAY
        return self._design.AssignDesignParameter(design_ref_id, design_param,
```

[docs]

```
    def AssignDesignCommand(self, design_ref_id: int, design_command_name: str,
        """
        Assign a design command to specified members in a design brief.

        Parameters
        ----------
        design_ref_id : int
            Design brief reference ID.
        design_command_name : str
            Name of the design command.
        design_command_value : str
```

```
            Value for the design command.
        member_ids : list of int
            List of member numbers.

        Returns
        -------
        int
            0 if successful, -1 otherwise.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ref_id = staad_obj.Design.CreateDesignBrief(1001)
        >>> result = staad_obj.Design.AssignDesignCommand(ref_id, "CHECK CODE",
        """
        if isinstance(member_ids, int):
            member_ids = [member_ids]
        safe_members = make_safe_array_long_input(member_ids)
        members_variant = make_variant_vt_ref(safe_members, automation.VT_ARRAY
        return self._design.AssignDesignCommand(design_ref_id, design_command_na
```

[docs]

```
    def AssignDesignGroup(self, design_ref_id: int, design_group_name: str, des
        """
        Assign physical members to a design group using a design command.

        Parameters
        ----------
        design_ref_id : int
            Design brief reference ID.
        design_group_name : str
            Name of the design group.
        design_group_value : str
            Value for the design group.
        same_as_member : int
            Reference member for the group.
        member_ids : list of int
            List of member numbers.

        Returns
        -------
        int
            0 if successful, -1 otherwise.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ref_id = staad_obj.Design.CreateDesignBrief(1001)
        >>> result = staad_obj.Design.AssignDesignGroup(ref_id, "scSteelGroup",
        """
        if isinstance(member_ids, int):
```

```
        member_ids = [member_ids]
    safe_members = make_safe_array_long_input(member_ids)
    members_variant = make_variant_vt_ref(safe_members, automation.VT_ARRAY
    return self._design.AssignDesignGroup(design_ref_id, design_group_name,
```

[docs]

```
def GetDesignBriefCode(self, design_ref_id: int):
    """
    Get the design code for a specified design brief.

    Parameters
    ----------
    design_ref_id : int
        Design brief reference ID.

    Returns
    -------
    int
        Design code.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> ref_id = staad_obj.Design.CreateDesignBrief(1001)
    >>> result = staad_obj.Design.GetDesignBriefCode(ref_id)
    """
    return self._design.GetDesignBriefCode(design_ref_id)
```

[docs]

```
def GetMemberDesignParameters(self, design_ref_id: int, member_no: int):
    """
    Get the design parameters for a specified member in a design brief.

    Parameters
    ----------
    design_ref_id : int
        Design brief reference ID.
    member_no : int
        Member number.

    Returns
    -------
    dict
        Dictionary containing:

        status : int
            Return code from COM (0 success, -1 failure or other).
        count : int | None
            Number of parameters available for the member (from COM object's
        _raw : COM object
            Original COM object (StaadPro.MembSteelDgnParams).
        parameters : dict[str, list]
            Mapping of parameter name -> [value, unit, description, default]
            unavailable it is set to None. Parameter names are returned exac

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> brief_id = 1
    >>> params = staad_obj.Design.GetMemberDesignParameters(brief_id, 1)
    """
    design_params = cc.CreateObject("StaadPro.MembSteelDgnParams")
    status = self._design.GetMemberDesignParameters(design_ref_id, member_no

    # Attempt to read count (number of parameters for the member)
    try:
        count = getattr(design_params, "Count")
    except Exception:
        count = None

    def _get_attr(name):
        try:
            return getattr(design_params, name)
        except Exception:
            return None

    # Raw COM attributes (may be SAFEARRAY, sequence, or callable indexer)
    raw_names = _get_attr("Name")
    raw_values = _get_attr("Value")
    raw_units = _get_attr("Unit")
    raw_descriptions = _get_attr("Description")
    raw_defaults = _get_attr("Default")

    def _materialize(obj):
        """Turn a COM collection or callable indexer into a Python list."""
        if obj is None:
            return []
        # Avoid splitting a single string into characters
        if isinstance(obj, str):
            return [obj]
        # Already a list/tuple
        if isinstance(obj, (list, tuple)):
            return list(obj)
        # Callable indexer pattern (obj(i)) if count known
        if callable(obj) and count is not None:
            result = []
            for i in range(int(count)):
                try:
                    result.append(obj(i))
                except Exception:
                    result.append(None)
            return result
        # Generic sequence protocol
        if hasattr(obj, "__len__") and hasattr(obj, "__getitem__"):
            try:
                return [obj[i] for i in range(len(obj))]
            except Exception:
                return []
        # Fallback: wrap scalar
        return [obj]

    names_list = _materialize(raw_names)
    values_list = _materialize(raw_values)
    units_list = _materialize(raw_units)
    descriptions_list = _materialize(raw_descriptions)
    defaults_list = _materialize(raw_defaults)

    parameters = {}
    for i, nm in enumerate(names_list):
        if nm is None:
            continue
        if isinstance(nm, str) and nm.strip() == "":
            continue
        parameters[str(nm)] = [
            values_list[i] if i < len(values_list) else None,
            units_list[i] if i < len(units_list) else None,
            descriptions_list[i] if i < len(descriptions_list) else None,
            defaults_list[i] if i < len(defaults_list) else None,
        ]

    return {
        "status": status,
        "count": count,
        "_raw": design_params,
        "parameters": parameters,
    }
```