Root

Contents

class openstaadpy.os_analytical.openstaadroot.OSRoot [source]

Bases: object

Analyze() [source]

Analyze the currently opened .STD file.

Return type:

None

> **See also**
>
> AnalyzeEx
>
> For more options.

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.Analyze()
```

AnalyzeEx(*silentMode: int, hiddenMode: int, waitTillComplete: int*)

Analyze the model with extended options. [source]

Notes

- This extended method analyzes the currently opened .STD file. This method is equivalent to running analysis from user interface.
- However, it has additional three arguments to specify whether to run the analysis in silent or hidden mode.
- The third parameter specifies whether the method should wait for the analysis to finish or return immediately.
- This method may be used in conjunction with SetSilentMode(), if one wants to suppress all dialog boxes displayed from the application during running of analysis.

Parameters:

- **silentMode** (*int*) – Integer value to enable silent mode. [1 = Enable, 0 otherwise]. Enabling silent mode will suppress all dialog boxes in the engine which requires user input. The analysis dialog box however will be displayed and close automatically on completion.
- **hiddenMode** (*int*) – Integer value to enable hidden mode. [1 = Enable, 0 = Disable]. Enabling hidden mode will suppress the display of analysis dialog. The analysis dialog box will not be displayed.
- **waitTillComplete** (*int*) – Integer value to specify whether to wait for the analysis process to finish or return immediately. [1 to wait , 0 otherwise]

Returns:

Returns -1 if Analysis Terminated. Returns 0 if General Error. Returns 1 if Analysis is in progress. Returns 2 if Analysis completed without errors or warnings. Returns 3 if Analysis completed with warnings but without errors. Returns 4 if Analysis completed with errors. Returns 5 if Analysis has not been performed.

Return type:

int

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> status = staad_obj.AnalyzeEx(1, 0, 1)
```

AnalyzeModel() [source]

Analyze the current model.

Return type:

None

See also

AnalyzeEx

For more options.

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.AnalyzeModel()
```

CloseSTAADFile() [source]

Close the currently open .STD file.

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.CloseSTAADFile()
```

GetAnalysisStatus(*modelPath: str = None*) [source]

Get analysis status for the open STAAD Model.

Parameters:

modelPath (*str, optional*) – The full path of the STAAD model. If not provided, the currently open model will be used.

Returns:

A dictionary containing the analysis status with following keys:

- **'ReturnValue' : *int***

  Status code of the analysis:

- **'ReturnString' : *str***

  Description corresponding to the *ReturnValue*.

- **'NoOfWarnings' : *int***

  Number of warnings generated during the analysis.

- **'NoOfErrors' : *int***

  Number of errors generated during the analysis.

- **'CPUTime' : *int***

  CPU time taken for the analysis in seconds.

Return type:

dict

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> status_dict = staad_obj.GetAnalysisStatus() # or staad_obj.GetAnaly
>>> print(f"Return Value: {status_dict['ReturnValue']}")
>>> print(f"Return String: {status_dict['ReturnString']}")
>>> print(f"No Of Warnings: {status_dict['NoOfWarnings']}")
>>> print(f"No Of Errors: {status_dict['NoOfErrors']}")
>>> print(f"CPU Time (sec): {status_dict['CPUTime']}")
```

GetApplicationVersion() [source]

Get the current application version as a string.

Returns:

The application version in the format 'X.X.X.X'.

Return type:

str

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> version = staad_obj.GetApplicationVersion()
>>> print(version)
```

GetBaseUnit() [source]

Get the base unit for the currently open .STD file.

- For English system of units (The values that are derived from a length unit, e.g. dimensions, areas, stresses, will be based on inches, 'in'. All values derived from a force unit, e.g. Axial force, moments, stresses, etc, will be based on kilopounds, 'KIP').

- For Metric system of units (The values that are derived from a length unit, will be based on Meters, ‘m’. All values derived from a force unit, will be based on kilo newtons, ‘kNs’).

Returns:

‘English’ or ‘Metric’.

Return type:

str

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> base_unit = staad_obj.GetBaseUnit()
>>> print(base_unit)
```

GetErrorMessage() [source]

Returns error messages thrown by OpenSTAAD (e.g. - for unavailability of license, unavailability of required named view)

Returns:

The error message string.

Return type:

str

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> msg = staad_obj.GetErrorMessage()
```

GetFullJobInfo() [source]

Get full job information for the current model.

Returns:

[job name, job client, engineer's name, engineer date, job number, revision, part name, reference, checker name, checker date, approver name, approval date, comments]

Return type:

list of 14 Strings

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> info = staad_obj.GetFullJobInfo()
```

GetInputUnitForForce() [source]

Retrieve the input unit of force of the currently open .STD file.

Returns:

The force unit name. ('Kilopound', 'Pound', 'Kilogram', 'Metric Ton', 'Newton', 'KiloNewton', 'MegaNewton', 'DecaNewton').

Return type:

str

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> force_unit = staad_obj.GetInputUnitForForce()
>>> print(force_unit)
```

GetInputUnitForLength() [source]

Retrieve the input unit of length of the currently open .STD file.

Returns:

The length unit name. (‘Inch’, ‘Feet’, ‘Feet’, ‘CentiMeter’, ‘Meter’, ‘MilliMeter’, ‘DeciMeter’, ‘KiloMeter’).

Return type:

str

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> length_unit = staad_obj.GetInputUnitForLength()
>>> print(length_unit)
```

GetMainWindowHandle() [source]

Get the main window handle of the STAAD.Pro application.

Returns:

Window handle.

Return type:

int

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> hwnd = staad_obj.GetMainWindowHandle()
```

GetProcessHandle() [source]

Retrieve the current STAAD.Pro process handle.

Returns:

The process handle.

Return type:

int

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> handle = staad_obj.GetProcessHandle()
>>> print(handle)
```

GetProcessId() [source]

Retrieve the current STAAD.Pro process ID.

Returns:

The process ID.

Return type:

int

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> pid = staad_obj.GetProcessId()
>>> print(pid)
```

GetSTAADFile(*bFullPath: bool = True*) [source]

Retrieve the path or the name of the current .STD file.

Parameters:

bFullPath (*bool, optional*) – If True, returns the full path. If False, returns only the file name. Default is True.

Returns:

The file path or name.

Return type:

str

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> file_path = staad_obj.GetSTAADFile() # or staad_obj.GetSTAADFile(bF
>>> print(file_path)
```

GetSTAADFileFolder() [source]

Retrieve the folder path of the current STAAD file.

Returns:

The folder path.

Return type:

str

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> folder = staad_obj.GetSTAADFileFolder()
>>> print(folder)
```

GetShortJobInfo() [source]

Get short job information for the current model.

Returns:

(job name, job ID, job status)

Return type:

tuple of 3 Strings

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> job_name, job_id, job_status = staad_obj.GetShortJobInfo()
>>> print(f"Job Name: {job_name}")
>>> print(f"Job ID: {job_id}")
>>> print(f"Job Status: {job_status}")
```

IsAnalyzing() [source]

Specify whether the analysis is running or not.

Returns:

True if analysis is running, False otherwise.

Return type:

bool

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.IsAnalyzing()
```

IsPhysicalModel()

[source]

Check if the current model is a physical model.

Return type:

bool

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.IsPhysicalModel()
```

NewSTAADFile(*fileName: str, lengthUnit: int, forceUnit: int*)

Create a .STD file with specified length and force units. [source]

Parameters:

- **fileName** (*str*) – The file name to save.
- **lengthUnit** (*int*) – Integer from 0 to 7 representing length unit (0-Inch, 1-Feet, 2-Feet, 3-CentiMeter, 4-Meter, 5-MilliMeter, 6-DeciMeter, 7-KiloMeter).
- **forceUnit** (*int*) – Integer from 0 to 7 representing force unit (0-Kilopound, 1-Pound, 2-Kilogram, 3-Metric Ton, 4-Newton, 5-Kilo Newton, 6-Mega Newton, 7-DecaNewton).

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.NewSTAADFile("C:/Models/new_model.std", 4, 5)
```

OpenSTAADFile(*file: str*) [source]

Open the specified .STD file.

Parameters:

file (*str*) – Path to the .STD file to open.

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.OpenSTAADFile("C:/Models/structure.std")
```

Quit() [source]

Close the STAAD.Pro application environment.

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.Quit()
```

SaveModel(*saveSilent: bool = False*) [source]

Save the current structure with optional silent mode.

Parameters:

saveSilent (*bool, optional*) – If True, saves the model silently. Default is False.

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.SaveModel()
>>> staad_obj.SaveModel(saveSilent=True)
```

SetFullJobInfo(*job_name: str, job_client: str = '', eng_name: str = '', eng_date: str = '', job_number: str = '', revision: str = '', part_name: str = '', reference: str = '', checker_name: str = '', checker_date: str = '', approver_name: str = '', approval_date: str = '', comments: str = ''*) [source]

Set full job information for the current model.

Parameters:

- **job_name** (*str*) – Name of the job.
- **job_client** (*str*) – Client name for the job.

- **eng_name** (*str*) – Engineer's name.
- **eng_date** (*str*) – Engineer's date.
- **job_number** (*str*) – Job number.
- **revision** (*str*) – Revision number.
- **part_name** (*str*) – Part name.
- **reference** (*str*) – Reference.
- **checker_name** (*str*) – Checker name.
- **checker_date** (*str*) – Checker date.
- **approver_name** (*str*) – Approver name.
- **approval_date** (*str*) – Approval date.
- **comments** (*str*) – Comments.

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.SetFullJobInfo("Full info")
```

SetInputUnitForForce(*forceUnit: int*) [source]

Set the input unit of force of the currently open .STD file.

Parameters:

forceUnit (*int*) – Integer from 0 to 7 representing force unit. (0- Kilopound, 1- Pound, 2- Kilogram, 3-Metric Ton, 4- Newton, 5-Kilo Newton, 6- Mega Newton, 7- DecaNewton).

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.SetInputUnitForForce(4) # Setting force unit to Newton
```

SetInputUnitForLength(*lengthUnit: int*) [source]

Set the input unit of length of the currently open .STD file.

Parameters:

lengthUnit (*int*) – Integer from 0 to 7 representing length unit. (0- Inch, 1- Feet, 2- Feet, 3- CentiMeter, 4- Meter, 5- MilliMeter, 6- DeciMeter, 7- KiloMeter).

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.SetInputUnitForLength(4)
```

SetInputUnits(*lengthUnit: int, forceUnit: int*) [source]

Set the input units of length and force of the currently open .STD file.

Parameters:

- **lengthUnit** (*int*) – Integer from 0 to 7 representing length unit. (0- Inch, 1- Feet, 2- Feet, 3- CentiMeter, 4- Meter, 5- MilliMeter, 6 - DeciMeter, 7 - KiloMeter).
- **forceUnit** (*int*) – Integer from 0 to 7 representing force unit. (0- Kilopound, 1- Pound, 2- Kilogram, 3-Metric Ton, 4- Newton, 5-Kilo Newton, 6- Mega Newton, 7- DecaNewton).

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.SetInputUnits(4, 5)
```

SetShortJobInfo(*job_name: str, job_id: str, job_status: str*)

Set short job information for the current model. [source]

Parameters:

- **job_name** (*str*) – Job name.
- **job_id** (*str*) – Job ID.
- **job_status** (*str*) – Job status.

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.SetShortJobInfo("NewJob", "12345", "In Progress")
```

SetSilentMode(*silent: bool*)

[source]

Set silent mode for the application.

Parameters:

silent (*bool*)

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.SetSilentMode(True)
```

ShowApplication() [source]

Bring the STAAD.Pro Application to the foreground.

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.ShowApplication()
```

UpdateStructure() [source]

Update the structure in the STAAD.Pro application.

Return type:

None

Examples

```
>>> from openstaadpy import os_analytical
>>> staad_obj = os_analytical.connect()
>>> if staad_obj is None:
...     print("staad object not found")
...     exit()
>>> staad_obj.UpdateStructure()
```

__init__(*filePath=None*) [source]

openstaadpy.os_analytical.openstaadroot.getActiveObject(*filePath: str = ''*) [source]