Step-by-step guide to using the OpenSTAAD API from Python

1. Install prerequisites

```
pip install comtypes pywin32 openstaad
```

2. Launch STAAD.Pro and connect

```
import subprocess, time, comtypes.client
from pythoncom import CoInitialize, CoUninitialize

CoInitialize()                       # Initialise COM
staad_path = r"C:\Program Files\Bentley\Engineering\STAAD.Pro 2024\STAAD\Bentley.Staad.exe"
subprocess.Popen([staad_path])
time.sleep(8)                        # Wait for STAAD to open

openstaad = comtypes.client.GetActiveObject("StaadPro.OpenSTAAD")
```

3. Create or open a model

```
from pathlib import Path
std_file_path = Path.cwd() / "my_model.std"
length_unit = 4   # 4 = metres
force_unit   = 5  # 5 = kN
openstaad.NewSTAADFile(str(std_file_path), length_unit, force_unit)
time.sleep(3)
```

4. Define material and section

```
prop = openstaad.Property
prop.SetMaterialName("STEEL")

# European IPE200 section
prop_no = prop.CreateBeamPropertyFromTable(
    country_code=7,        # 7 = European database
    section_name="IPE200",
    type_spec=0,           # single section from table
    add_spec_1=0.0,
    add_spec_2=0.0
)
```

5. Add nodes and beams

```
geom = openstaad.Geometry
geom.CreateNode(1, 0, 0, 0)
geom.CreateNode(2, 5, 0, 0)
geom.CreateBeam(1, 1, 2)          # Beam 1: node 1 → node 2
prop.AssignBeamProperty(1, prop_no)
```

6. Supports and loads

```
sup = openstaad.Support
sup_no = sup.CreateSupportFixed()
sup.AssignSupportToNode(1, sup_no)
sup.AssignSupportToNode(2, sup_no)

ld = openstaad.Load
case = ld.CreateNewPrimaryLoad("Self-Weight")
ld.SetLoadActive(case)
ld.AddSelfWeightInXYZ(case, -1.0)   # factor −1 in global Y
```

7. Run the analysis (silent mode)

```
cmd = openstaad.Command
cmd.PerformAnalysis(6)      # 6 = static analysis
```

```
openstaad.SetSilentMode(1)
openstaad.Analyze()
while openstaad.isAnalyzing():
    time.sleep(2)
```

8. Retrieve results

```
from openstaad import Output
out = Output()
fx, fy, fz, mx, my, mz = out.GetMemberEndForces(beam=1, start=True, lc=1)
print("Start-end forces:", fx, fy, fz, mx, my, mz)
```

9. Clean up

```
openstaad.SaveModel(1)
CoUninitialize()
```

10. Helper wrappers

If you prefer a higher-level interface, install **OpenStaadPython**:

```
pip install openstaad
```

Then use convenience classes:

```
from openstaad import Geometry, Root
print(Geometry().GetBeamList())
print(Root().GetSTAADFile())
```

(Note: `openstaad` currently focuses on **querying** an **already-open** model.)

Documentation & Community

- **Official docs**:
  `C:\Program Files\Bentley\Engineering\STAAD.Pro 2024\OSAPP_Help`
  (Examples are mainly VB/C++; Python help lives in the Bentley forums.)
- **GitHub samples**:
  viktor-platform/sample-staad-integration

You can now create, modify, analyse and extract results from STAAD.Pro entirely from Python scripts.

Revision #1
Created 18 July 2025 00:10:02 by Admin
Updated 18 July 2025 00:12:32 by Admin