PyEnSight cheat sheet

Version: 0.5.1 (stable)

Launch EnSight

Launch EnSight locally:

```
from ansys.pyensight.core import launch_ensight

install = "C:\\Program Files\\ANSYS Inc\\v232"
session = launch_ensight(ansys_installation=install)
```

Launch EnSight locally using the launcher:

```
from ansys.pyensight.core import LocalLauncher

install = "C:\\Program Files\\ANSYS Inc\\v232"
session = LocalLauncher(
    ansys_installation=install,
    batch=False, # Launch interactive GUI
).start()
```

Launch EnSight locally from a container:

```
from ansys.pyensight.core import DockerLauncher

launcher = DockerLauncher(data_directory="d:/data")
launcher.pull()
session = launcher.start()
```

Load data

Load an example for postprocessing a design point:

```
session.load_example("elbow_dp0_dp1.ens")
```

Load EnSight, Fluent, or Mechanical data:

```
session.load_data("dataset.encas")
session.load_data("dataset.dvs")
session.load_data("out.cas.h5", result_file="data.dat.
    h5")
session.load_data("dataset.rst", new_case=True)
# The "new_case" option allows you to create a new
    case and load multiple datasets in the same
    EnSight session.
```

PyEnSight renderables

Create an image renderable:

```
image = session.show("image")
# "image" is a renderable object. Any object returned
    by the "show()" method is a renderable.
image.browser()
```

Create a deep-pixel picture, an animation, and a WebGL renderable:

```
session.show("deep_pixel")
session.show("webgl")
session.show("animation")
```

Display the EnSight rendering window:

```
session.show("remote").browser()
session.show("remote").url
# The "url" property is the URL for embedding the
    renderable in your application. It is available
    in all renderables, like ``browser()``.
```

Working with EnSight

Export a picture, a deep-pixel picture, and an animation:

```
export = session.ensight.utils.export
export.image("image.png")
export.image("deep_image.tiff", enhanced=True)
export.animation("animation.mp4")
```

Select parts by a specific dimension:

```
parts = session.ensight.utils.parts
parts.select_parts_by_dimension(3) # 3D parts
```

Set an isometric view:

```
views = session.ensight.utils.views
views.set_view_direction(1, 1, 1)
```

Query a 1D part and plot the result:

```
query = session.ensight.utils.query
sn = session.ensight.utils.support.scoped_name
with sn(session.ensight.objs.core) as core:
    query.create_distance(
        "my_query",
        query.DISTANCE_PART1D,
        [ensight_1D_object], # ENS_PART
        core.VARIABLES["my_variable"][0],
        new_plotter=True,
    )
```

Saving and restoring

Save in memory:

```
state = session.capture_context()
```

Save on disk:

```
state.save("state_on_file.ctxz")
```

Restore from memory:

```
session.restore_context(state)
```

Restore from disk:

```
new_ctx = EnsContext()
new_ctx.load("state_in_file.ctxz")
session.restore_context(new_ctx)
```

Common graphics operations

Color by a variable:

```
attr = "COLORBYPALETTE"
sn = session.ensight.utils.support.scoped_name
with sn(session.ensight.objs.core) as core:
    core.PARTS.set_attr(attr, "velocity")
    core.PARTS[0].setattr(attr, "energy")
    core.PARTS["fluid_domain"].set_attr(attr, "tke")
# The "set_attr()" method is available for a list of
    EnSight objects, while the "setattr()" method is
    available for a single object.
```

Change the representation:

```
sn = session.ensight.utils.support.scoped_name
with sn(session.ensight.objs.core) as core, sn(
    session.ensight.objs.enums
) as enums:
    # 3D parts border, 2D full
    core.PARTS.set_attr("ELTREPRESENTATION", enums.
        BORD_FULL)
    # High-quality smooth shading
    core.PARTS.set_attr("SHADING", enums.
        SHAD_SMOOTH_REFINED)
```

Show the mesh lines:

```
session.ensight.objs.core.HIDDENLINE = True
session.ensight.objs.core.HIDDENLINE_USE_RGB = True
session.ensight.objs.core.HIDDENLINE_RGB = [0.0, 0.0,
    0.0]
```

References from PyEnSight documentation

- Getting started
- EnSight API through PyEnSight
- Examples