PyAEDT-API cheat sheet

Version: 0.6.78 (stable)

Launch PyAEDT

Launch an HFSS instance locally:

```
import pyaedt
hfss = pyaedt.Hfss(specified_version="2023.1",
        non_graphical=False,
        new_desktop_session=True,
        projectname="Project_name",
        designname="Design_name")
```

Exit your local instance:

```
hfss.release_desktop()
```

Variable class

The `hfss.variable_manager` class handles all variables.
Create a variable that only applies to this design:

```
hfss["dim"] = "1mm"
```

Create a variable that applies at a project level:

```
hfss["$dim"] = "1mm"
```

Material class

The `hfss.materials` class is used to access the materials library.
Add a new material:

```
my_mat = hfss.materials.add_material("myMat")
my_mat.permittivity = 3.5
my_mat.conductivity = 450000
my_mat.permeability = 1.5
```

Geometry creation

The `hfss.modeler` class contains all methods and properties needed to edit a modeler, including those for primitives.

Draw a box at (x_pos, y_pos, z_pos) position with (x_dim, y_dim, z_dim) dimensions:

```
box = hfss.modeler.create_box([x_pos,y_pos,z_pos], [
    x_dim,y_dim,z_dim],name="airbox", matname="air")
```

Create a spiral geometry made of copper:

```
ind = hfss.modeler.create_spiral(
        internal_radius=rin, width=width,
        spacing=spacing, turns=Nr,
        faces=Np, thickness=thickness,
        material="copper",name="Inductor1")
```

The `Object3d objects`, `box` and `ind`, contain a lot of methods and properties related to that object, including faces, vertices, colors, and materials.

Boundary creation

Create an open region:

```
hfss.create_open_region(Frequency="1GHz")
```

Assign a radiation boundary:

```
hfss.assign_radiation_boundary_to_objects("airbox")
```

Port definition

Common port types in HFSS are lumped port and wave port.

Define a lumped port:

```
box1 = hfss.modeler.create_box([0,0,50], [10,10,5], "
    Box1","copper")
box2 = hfss.modeler.create_box([0,0,60], [10,10,5], "
    Box2","copper")
port_L = hfss.lumped_port(signal="Box1", reference="
    Box2", integration_line=hfss.AxisDir.XNeg,
    impedance=50, name="LumpedPort", renormalize=True
    , deembed=False)
```

Define a wave port:

```
port_W = hfss.wave_port("Box1", "Box2", name="WavePort
    ", integration_line=1)
```

Setup class

The `hfss.create.setup` class is used to define the solution setup:

```
setup = hfss.create_setup("MySetup")
setup.props["Frequency"] = "50MHz"
setup["MaximumPasses"] = 10
hfss.create_linear_count_sweep(setupname="any", unit="
    MHz", freqstart=0.1, freqstop=100,
    num_of_freq_points=100, sweepname="sweep1",
    sweep_type="Interpolating", save_fields=False)
```

Access the parametric sweep:

```
hfss.parametrics
```

Access the optimizations:

```
hfss.optimizations
```

Mesh class

The `hfss.mesh` module manages the mesh functions:

```
hfss.mesh.assign_initial_mesh_from_slider(level=6) #
    Set the slider level to 6
# Assign model resolutions
hfss.mesh.assign_model_resolution(names=[object1.name,
    object2.name], defeature_length=None)
# Assign mesh length to \texttt{object1} faces
hfss.mesh.assign_length_mesh(names=object1.faces,
    isinside=False, maxlength=1, maxel=2000)
```

Analyze class

The `analyze` class is used to analyze a solution setup (`mysetup`) in an HFSS design:

```
hfss.analyze_setup("mysetup")
```

Post class

The `post` class has methods for creating and editing plots in AEDT:

```
plotf = hfss.post.create_fieldplot_volume(object_list,
    quantityname, setup_name, intrinsic_dict) # This
    call returns a FieldPlot object
my_data = hfss.post.get_solution_data(expression=
    trace_names) # This call returns a SolutionData
    object
standard_report = hfss.post.report_by_category.
    standard("db(S(1,1))") # This call returns a new
    standard report object
standard_report.create() # This call creates a report
solution_data = standard_report.get_solution_data()
```

Call AEDT-API with PyAEDT

Most core functionality can be called directly through PyAEDT, but additional features can be added by converting the corresponding AEDT-API methods.

For example, access the `Optimetrics` module:

```
omodule = hfss.odesign.GetModule("Optimetrics")
```

References from PyAEDT documentation

- Getting started
- User guide
- API reference