PyMAPDL cheat sheet

Version: 0.65.0 (stable)

/ Launch PyMAPDL

Launch a PyMAPDL instance locally and exit it:

```
# Launch an instance
from ansys.mapdl.core import launch_mapdl
mapdl = launch_mapdl()
# Exit the instance
mapdl.exit()
```

Specify a job name, number of processors, and working directory:

```
jname = 'user_jobname'
path = '<path of directory>'
mapdl = launch_mapdl(nproc=2, run_location=path,
    jobname=jname)
```

Connect to an existing instance of MAPDL at IP address 192.168.1.30 and port 50001:

```
mapdl = launch_mapdl(start_instance=False,
    ip='192.168.1.30', 50001)
```

Create and exit a pool of instances:

```
# Create a pool of 10 instances
from ansys.mapdl.core import LocalMapdlPool
pool = mapdl.LocalMapdlPool(10)
# Exit the pool
pool.exit()
```

/ PyMAPDL commands

PyMAPDL commands are Python statements that act as a wrapper for APDL commands. For example, `ESEL, s, type, 1` is translated as:

```
mapdl.esel('s', 'type', vmin=1)
```

Commands that start with * or / have these characters removed:

```
mapdl.prep7()           # /PREP7
mapdl.get()             # *GET
```

In cases where removing * or / causes conflicts with other commands, a prefix "slash" or "star" is added:

```
mapdl.solu()            # SOLU
mapdl.slashsolu()       # /SOLU

mapdl.vget()            # VGET
mapdl.starvget()        # *VGET
```

Convert an existing APDL script to PyMAPDL format:

```
inputfile = 'ansys_inputfile.inp'
pyscript = 'pyscript.py'
mapdl.convert_script(inputfile, pyscript)
```

/ MAPDL class

Load a table from Python to MAPDL:

```
mapdl.load_table(name, array)
```

Access from or write parameters to the MAPDL database:

```
# Save a parameter to a NumPy array
nparray = mapdl.parameters['displ_load']
# Create a parameter from a NumPy array
mapdl.parameters['exp_disp'] = nparray
```

Access information using *GET and *VGET and store it in NumPy arrays:

```
# Run *GET command and return a Python value
mapdl.get_value(entity='NODE', item1='COUNT')

# Run *VGET command and return a Python array
mapdl.get_array(entity='NODE', item1='NLIST')
```

/ Mesh class

Store the finite element mesh as a VTK `UnstructuredGrid` data object:

```
grid = mapdl.mesh.grid
```

Save element and node numbers to Python arrays:

```
# Get an array of the nodal coordinates
nodes = mapdl.mesh.nodes

# Save node numbers of selected nodes to an array
node_num = mapdl.mesh.nnum
# Save node numbers of all nodes to an array
node_num_all = mapdl.mesh.nnum_all

# Get element numbers of currently selected elements
elem_num = mapdl.mesh.enum
# Get all element numbers, including those not
    selected
elem_num_all = mapdl.mesh.enum_all
```

/ Post-processing class

The `post_processing` class is used for plotting and saving results to NumPy arrays.

```
from ansys.mapdl.core import post_processing
mapdl.post1()
mapdl.set(1, 2)
# Plot the nodal equivalent stress
mapdl.post_processing.plot_nodal_eqv_stress()
# Save nodal equivalent stresses to a Python array
nod_eqv_stress = post_processing.nodal_eqv_stress()
# Plot contour legend using dictionary
mapdl.allsel()
sbar_kwargs = {"color": "black",
               "title": "Equivalent Stress (psi)",
               "vertical": False,
               "n_labels": 6}
post_processing.plot_nodal_eqv_stress(
    cpos='xy', background='white',
    edge_color='black', show_edges=True,
    scalar bar_args=sbar_kwargs, n_colors=9)
```

/ Plotting class

Use PyVista to interpolate data, saving the resulting stress and storing it in the underlying `UnstructuredGrid` object:

```
pl = pyvista.Plotter()
pl0 = post_processing.plot_nodal_stress(
    return_plotter=True)
pl.add(pl0.mesh)
pl.show()
```

```
# Plot currently selected elements
mapdl.eplot(show_node_numbering, vtk)
# Plot selected volumes
mapdl.vplot(nv1, nv2, ninc, degen, scale, ...)
# Display selected areas
mapdl.aplot(na1, na2, ninc, degen, scale, ...)
# Display selected lines without MAPDL plot symbols
mapdl.lplot(vtk=True, cpos='xy', line_width=10)
# Save PNG file of line plot with MAPDL coordinate
    symbol
mapdl.psymb('CS', 1)
mapdl.lplot(vtk=False)
```

References from PyMAPDL documentation

- Getting started
- MAPDL commands
- API reference