Cheat sheet for PyMAPDL

Launching PyMAPDL

To launch PyMAPDL instance locally and exit it

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

To specify a jobname, number of processors, and working directory

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

To connect to an existing instance of MAPDL at IP 192.168.1.30 and port 50001.

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

To create and exit a pool of instances

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

PyMAPDL Language

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

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

Commands that start with * or / have those characters removed.

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

In cases where removing * or / will cause conflict with other commands, a prefix "slash" or "star" is added.

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

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

Converting 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,var1='', var2='',
                 var3='', csysid='')
```

To access from or write parameters to MAPDL database

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

To access information using *GET and *VGET directly to NumPy arrays

```
# Runs *GET command and returns a Python value
mapdl.get_value(entity='', entnum='',
                item1='', it1num='',
                item2='', it2num='', **kwargs)

# Runs *VGET command and returns a Python array
mapdl.get_array(entity='', entnum='',
                item1='', it1num='', item2='',
                it2num='', kloop='', **kwargs)
```

Mesh Class

Store the finite element mesh as a VTK UnstructuredGrid data object.

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

Save element & node numbers to Python arrays.

```
# Array of nodal coordinates
nodes=mapdl.mesh.nodes

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

# Element numbs. of currently selected elements
elem_num=mapdl.mesh.enum
# All element numbers incl. those not selected
elem_num_all=mapdl.mesh.enum_all
```

Post-Processing Class

This class is used for plotting and saving results to NumPy arrays.

```
mapdl.post1()
mapdl.set(1, 2)
# Plot the nodal equivalent stress
mapdl.post_processing.plot_nodal_eqv_stress()
# Save nodal eqv. stresses to a Python array
nod_eqv_stress=
        mapdl.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}
mapdl.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

Plotting is interpolated with PyVista by saving the resulting stress and storing wtihin the underlying UnstructuredGrid

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

```
# Plot the currently selected elements
mapdl.eplot(show_node_numbering, vtk)
# Plot the selected volumes
mapdl.vplot(nv1, nv2, ninc, degen, scale, ...)
# Display the selected areas
mapdl.aplot(na1, na2, ninc, degen, scale, ...)
# Display the 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