Cheatsheet 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, num. of processors, and working directory

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

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 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

```
# To save a parameter named 'displ_load' to a
    NumPy array nparray
nparray=mapdl.parameters['displ_load']
# To create a parameter named 'exp_disp' from a
    NumPy array nparray
mapdl.parameters['exp_disp']=nparray
```

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

```
# Runs the *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.

```
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 selected nodes to array
node_num_all=mapdl.mesh.nnum_all

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

Post-processing Class

To plot results the general form is:
`mapdl.postprocessing.result_name`

```
mapdl.post1()
mapdl.set(1, 2)
# To plot the nodal equivalent stress
mapdl.post_processing.plot_nodal_eqv_stress()

# To save nodal eqv. stresses to a Python array
nod_eqv_stress=mapdl.post_processing.
    nodal_eqv_stress()
```

Plotting Class

MAPDL Math Class

References

- Getting Started
- MAPDL Commands
- API Reference