PyMechanical cheat sheet

Version: main

Connect to Mechanical Remote session

Launch and connect to session

```
# Launch an instance
from ansys.mechanical.core import launch_mechanical
mechanical = launch_mechanical()
```

Launch Mechanical UI

```
mechanical = launch_mechanical(batch=False)
```

Launch by version

Verify the license and version of Mechanical that is used:

```
print(mechanical)
```

Launch a specific version of Mechanical:

```
from ansys.mechanical.core import find_mechanical
wb_exe = find_mechanical(251)[0]
# 'Ansys Inc\\....\\winx64\\AnsysWBU.exe'
mechanical = launch_mechanical(
  exec_file=wb_exe, verbose_mechanical=True,
  batch=True
  )
print(mechanical)
```

Launch Mechanical using the CLI

```
ansys-mechanical -r 251 --port 10000 -g
```

Manually connect to the Mechanical session

```
from ansys.mechanical.core import
    connect_to_mechanical
# Connect locally
mechanical = connect_to_mechanical(port=10000)
          # Or
# Connect remotely, to the IP address or hostname
mechanical = connect_to_mechanical(
  "192.168.0.1", port=10000
  )
```

Send commands to Mechanical

Run a single command:

```
result1 = mechanical.run_python_script("2+3")
result2 = mechanical.run_python_script(
    "DataModel.Project.ProjectDirectory"
)
mechanical.run_python_script(
    "Model.AddStaticStructuralAnalysis()"
)
```

Evaluate a block of commands:

```
# Import a material
commands = """
cu_mat_file_path = (
    r'D:\Workdir\copper.xml'.replace("\\", "\\\\")
)
materials = Model.Materials
materials.Import(cu_mat__file_path)
"""
mechanical.run_python_script(commands)
```

Execute a Python script:

```
mechanical.run_python_script_from_file(file_path)
```

Import a Mechanical file and print the count of bodies:

```
file = r"D:\\Workdir\\bracket.mechdb"
command = f'DataModel.Project.Open("{file}")'
mechanical.run_python_script(command)
mechanical.run_python_script("""
allbodies = Model.GetChildren(
    DataModelObjectCategory.Body, True)
""")
mechanical.run_python_script("allbodies.Count")
```

Perform project-specific operations:

```
# Get the project directory
mechanical.project_directory
# List the files in the working directory.
mechanical.list_files()
# Save
mechanical.run_python_script(
"ExtAPI.DataModel.Project.Save(r'D:\\Workdir')")
# Log in two ways:
mechanical._log.info("This is a useful message.")
mechanical.log_message("INFO", "info message")
# Exit
mechanical.exit(force=True)
```

Load a Mechanical embedded instance

Start an instance of App

```
from ansys.mechanical.core import App
app = App(version=251)
print(app)
```

Extract and merge global API entry points:

```
# Extract global API entry points from Mechanical
# Merge them into your Python global variables
app.update_globals(globals())
```

Access entry points from Python:

```
ExtAPI # Application.ExtAPI
DataModel # Application.DataModel
Model # Application.DataModel.Project.Model
Tree # Application.DataModel.Tree
Graphics # Application.ExtAPI.Graphics
```

Import a file and print the count of bodies

```
file = r"D:\\Workdir\\bracket.mechdb"
app.open(file)
app.update_globals(globals())
allbodies = Model.GetChildren(
  DataModelObjectCategory.Body, True)
print(allbodies.Count)
```

Turn on warning logging:

```
import logging
from ansys.mechanical.core import App
from ansys.mechanical.core.embedding.logger
import Configuration,Logger
Configuration.configure(level=logging.WARNING,
to_stdout=True)
app = App(version=251)
Logger.error("Test Error Message")
```

Visualize geometry in 3D:

```
# requires Mechanical version >= 24R2
app.plot()
```

Print project structure as tree:

```
app.print_tree()
# print only 20 lines
app.print_tree(max_lines=20)
```