PyAnsys: Revolutionizing Engineering Through the Power of Python

Rajesh Meena

Mehul Muradia

Ansys

Who are we?

Rajesh Meena
Sr. Technology Specialist
Ansys Pune

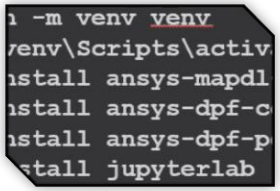
Mehul Muradia
Technology Specialist
Ansys Pune

- Automation and Customization of Ansys Products
- Customized Workflow Implementations
- Structural Simulation Experts

Agenda

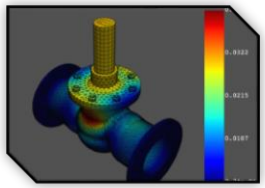
What is PyAnsys?

Why use it?



Getting Started

- MAPDL
  - From MAPDL to PyMAPDL
- Mech
  - PyMechanical
- DPF
  - What's DPF
  - PyDPF (post and core)
- Other libraries
  - PyPrimeMesh
  - PySherlock etc



Libraries and demos

Final remarks + Q&A

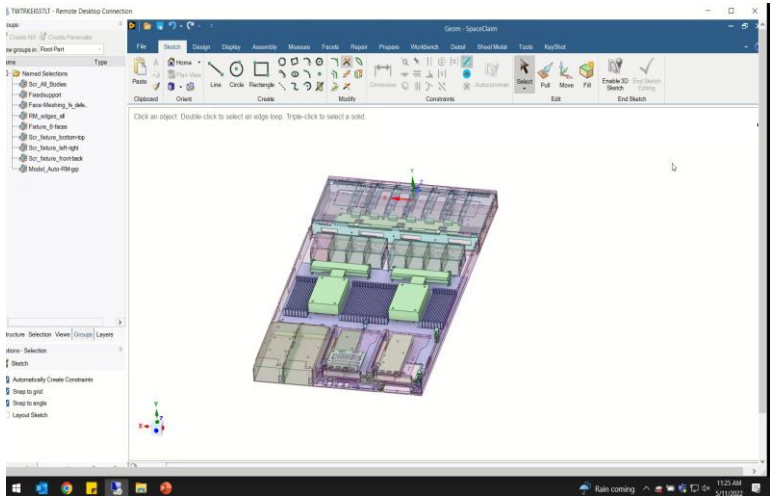
What Should be the takeaway?

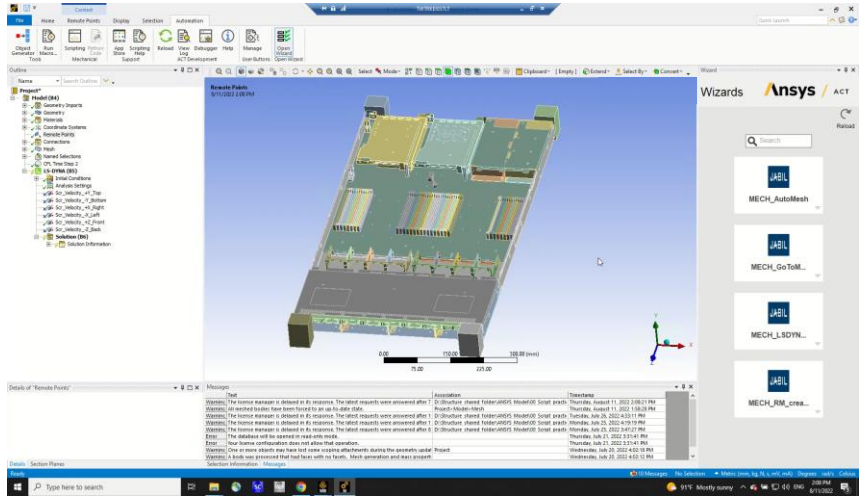
How does **PyAnsys** makes difference..?

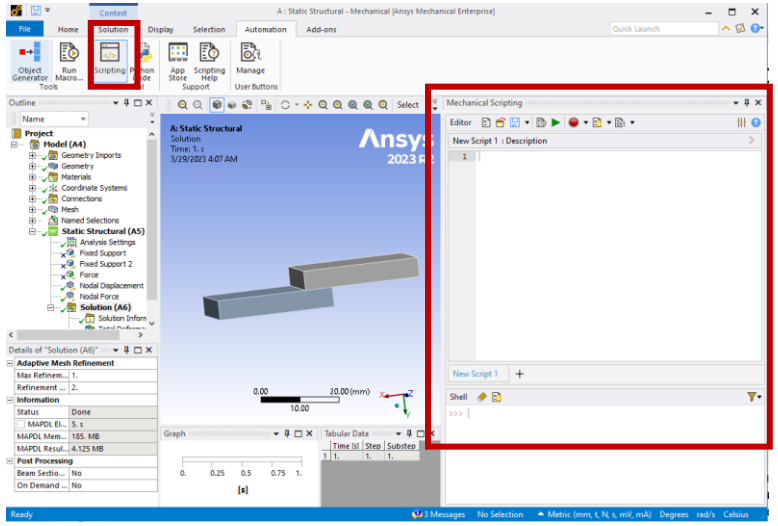
How can I get started **myself**?

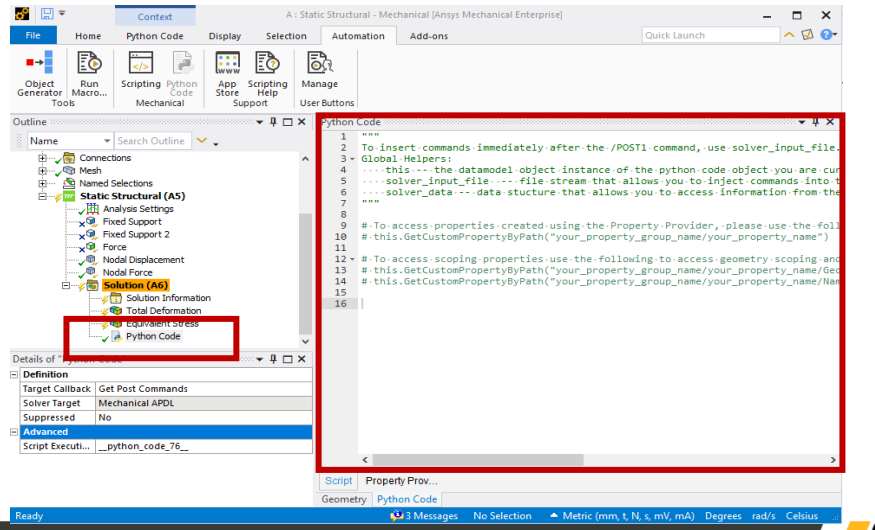
Traditional Automation in Ansys (In-Tool)

In-tool Automation









In-tool Automation

- Good for:
  - customizing the software (ACT)
  - Automate repetitive manual operations
- Not optimal for:
  - Muti-tool workflow automation
  - Fully Automated workflow, with maintainability
  - Use of 3rd party tool in the workflow
  - Train AI pair programmer

Simulation Democratization: Facts, Challenges, Solution

Facts	Challenges	Solution: Python
100s Simulation Tools	OS Dependency	Platform Independent
1000s Workflows/Scripts	Version Management	Wider Adoption
10s Programming Languages	Full Scale Automation	Large Community Support
	Maintenance/Upgrade	Technology Enabler
	Interact with Data Packages	Pre-Built Libraries
	Deployment	APIs

What’s PyAnsys?

What is PyAnsys?

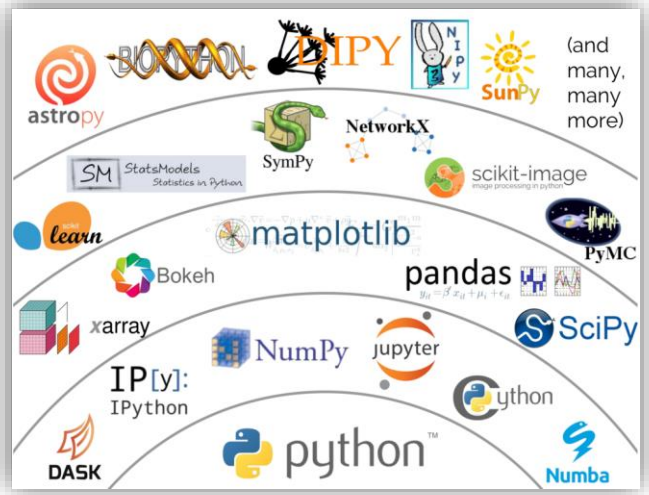
Set of technologies that allow the user to interface with Ansys products pythonically

python™ + [Ansys] = py[Ansys]

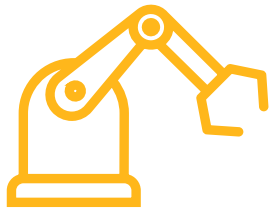
Python | **Ansys** | **PyAnsys**

The PyAnsys project is **Ansys's commitment to open-source** where we provide Python libraries that expose Ansys technologies in the Python ecosystem through APIs and interfaces that are **clear**, **concise**, and **maintainable**. This allows Ansys customers to do:

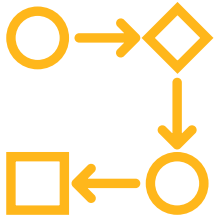
- **Flexible Automation:** Democratize powerful capabilities offered by Ansys through scripting
- **Flexible Distribution :** Connect Ansys and Open-Source technologies in a seamless manner
- **Broader Technology Integration:** Integrate Ansys physics capabilities easily with AI/ML



PyAnsys: Ansys's Commitment to Open Source



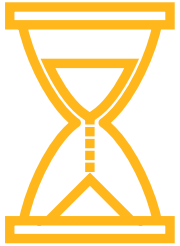
Better Automation



Simple Workflow

Access from Anywhere

Integration with AI/ML

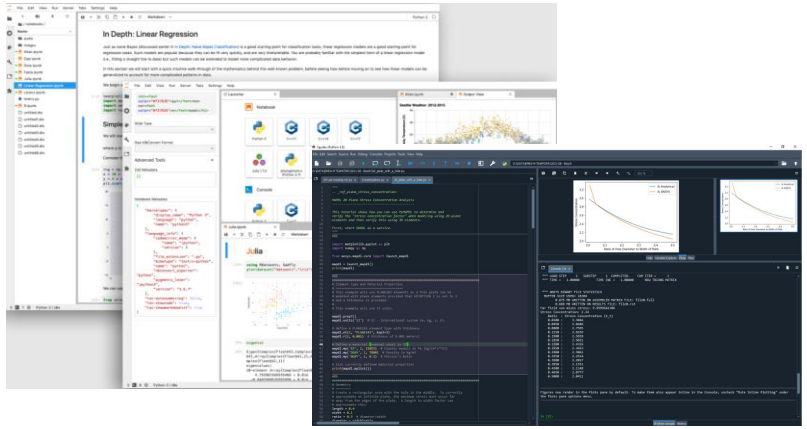


Real Time Post Process

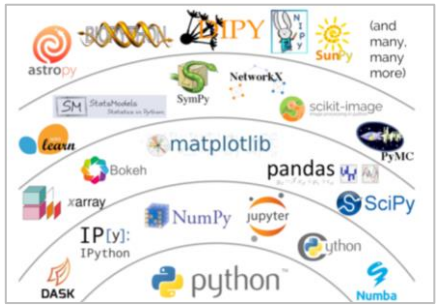
- Ansys Technologies in Python Ecosystem
- Clear, Concise, Maintainable APIs
- Flexible Automation
- Flexible Distribution
- Broader Technology Integration
- Easier for Deployment

Scripting from within the customer tools – PyAnsys

Any tool from which you can write Python (UI is yours to choose)



Possibility to use other useful Python tools



Customer tools

PyAnsys technology

Products

API — Ansys Product A

API — Ansys Product B

API — Ansys MAPDL

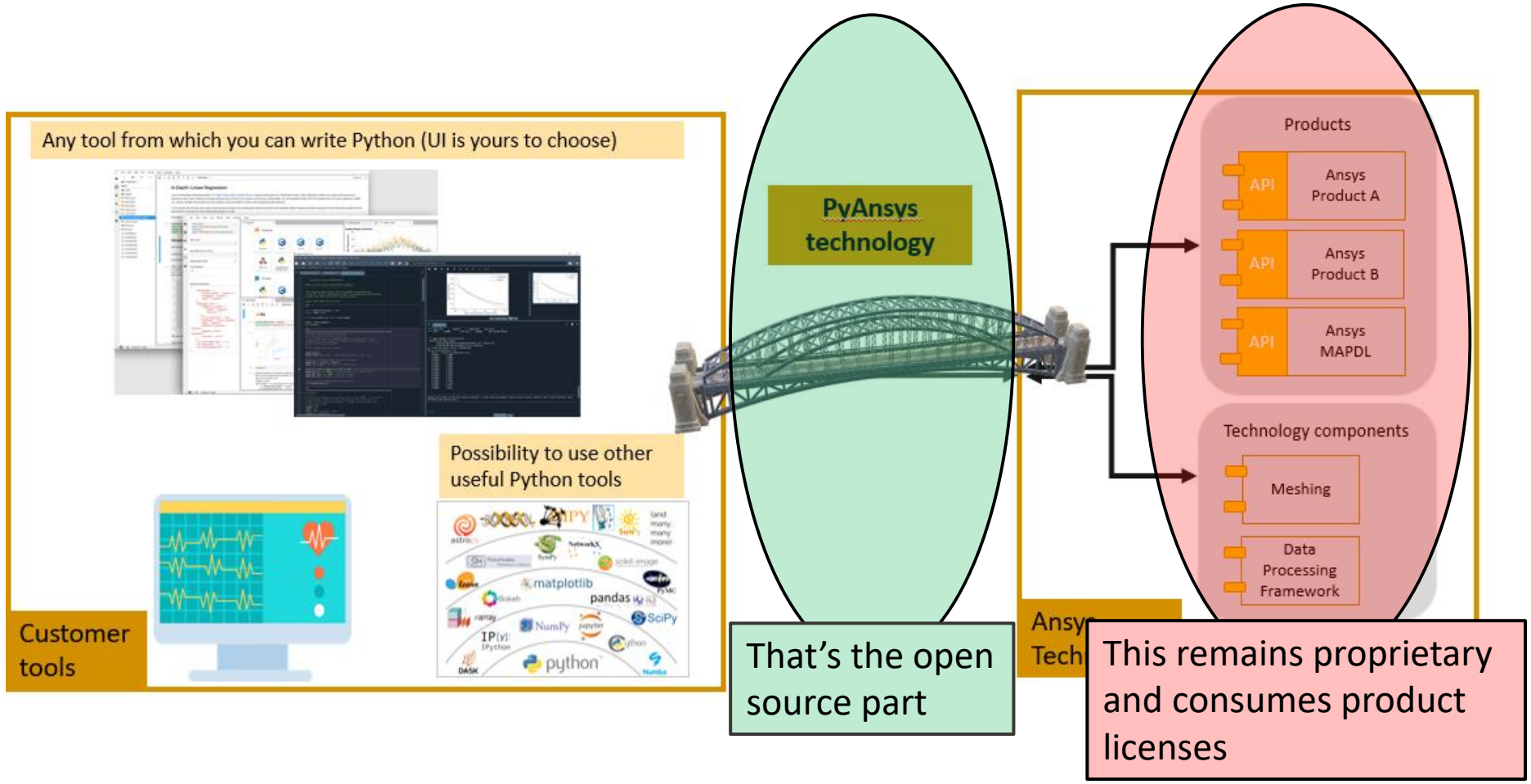
Technology components

Meshing

Data Processing Framework

Ansys Technology

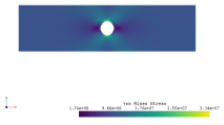
Open source ?! What's in it for us?



We're just offering a new way of using our products!
This will allow existing users to use our products differently, effectively and efficiently.

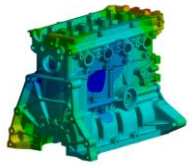
Status of PyAnsys?

Simulation libraries



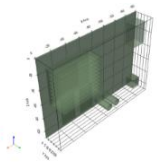
PyMAPDL

Pythonic interface to Ansys MAPDL (Mechanical APLD)



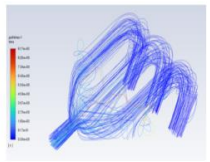
PyMechanical

Pythonic interface to Ansys Mechanical



PyAEDT

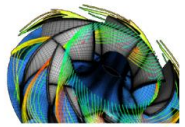
Pythonic interface to AEDT (Ansys Electronic Desktop)



PyFluent

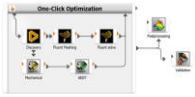
Pythonic interface to Ansys Fluent

Utility libraries



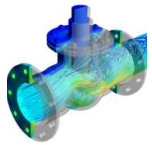
PyPrimeMesh

Pythonic interface to Ansys Prime Server, which delivers core Ansys meshing technology



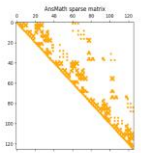
PyOptislang

Pythonic interface to Ansys Optislang



PySystem Coupling

Pythonic interface to communicate with Ansys System Coupling



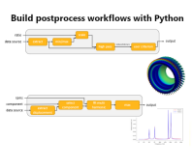
PyAnsys Math

Pythonic interface to PyAnsys Math libraries

PyDynamicReporting

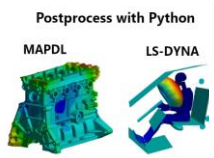
Pythonic interface to Ansys Dynamic Reporting for service and control of its database and reports

Post-processing libraries



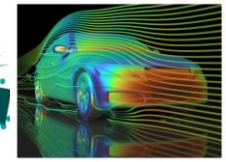
PyDPF - Core

Pythonic interface to DPF (Data Processing Framework) for building more advanced and customized workflows



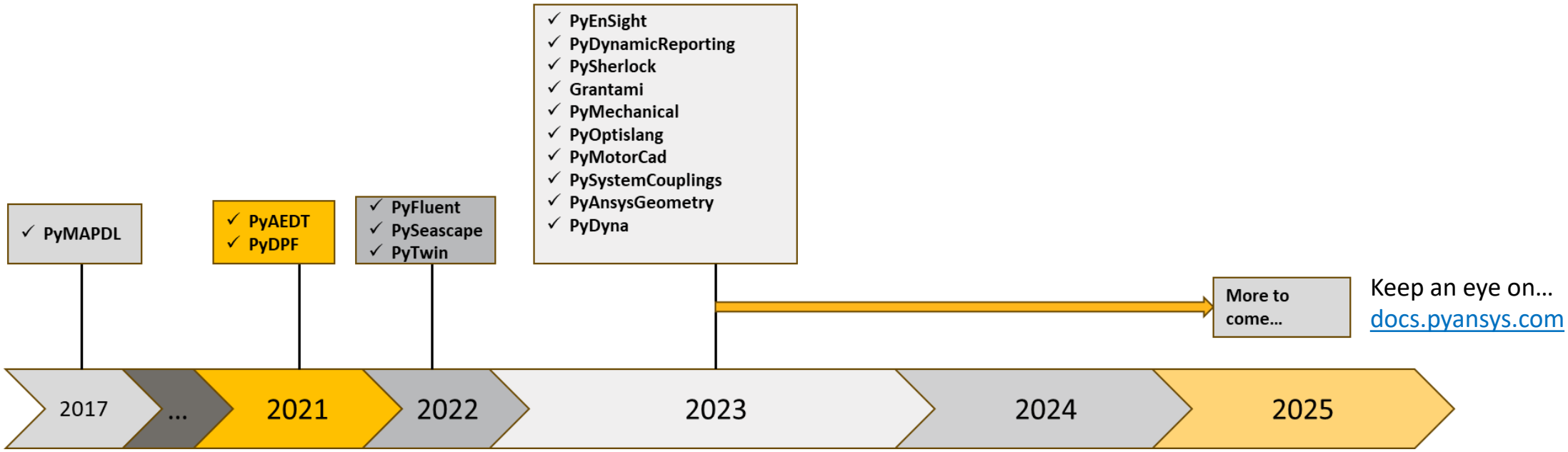
PyDPF - Post

Pythonic interface to access and post process Ansys solver result files



PyEnSight

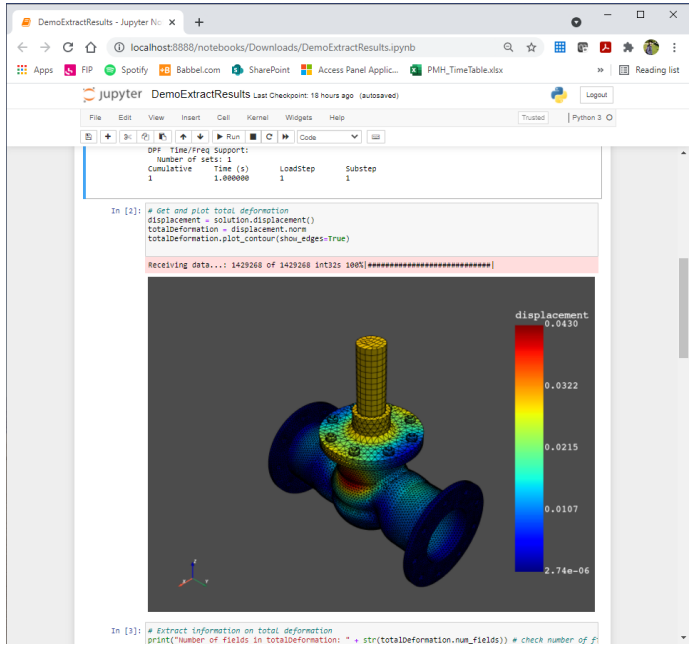
Pythonic interface to EnSight, the Ansys simulation postprocessor



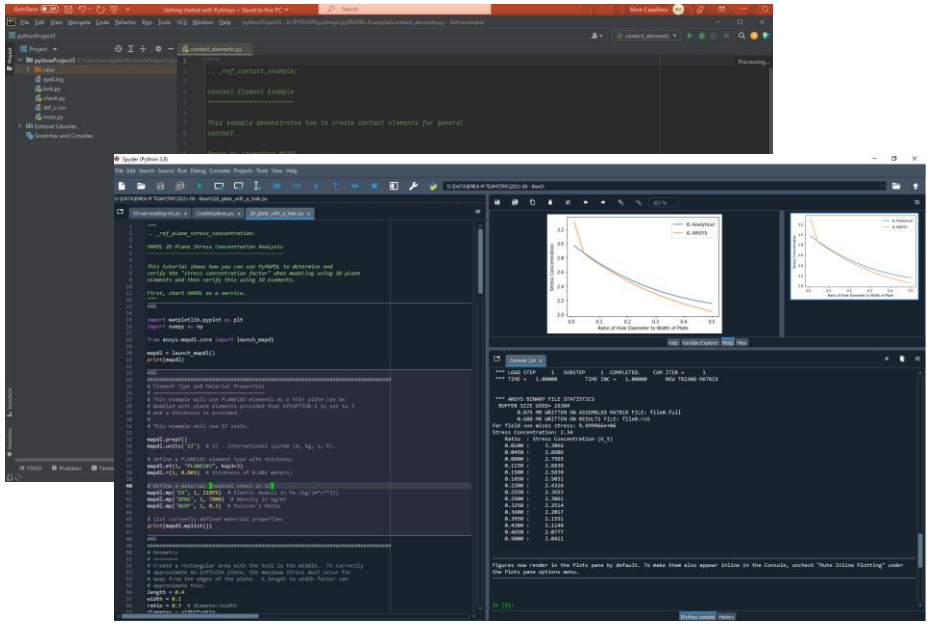
Keep an eye on…
docs.pyansys.com

What's the UI like?

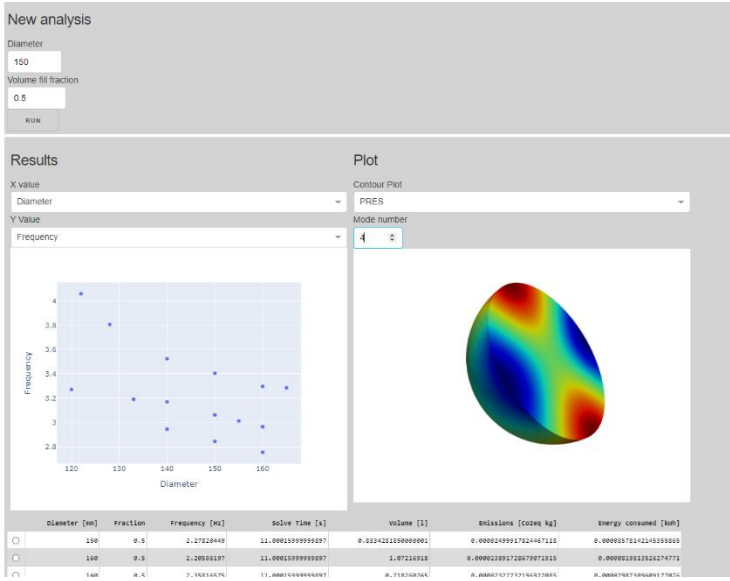
- PyAnsys is a set of tools to interact with Ansys products from a Python interface
- Any Python interface can be used (even the Windows Command Prompt !)
- Using an IDE (Integrated Development Environment) is recommended to write and maintain code.
  Many IDEs are available: Spyder, PyCharm, VS Code, Visual Studio, ….



Code can be written from a web-page thanks to Jupyterlab



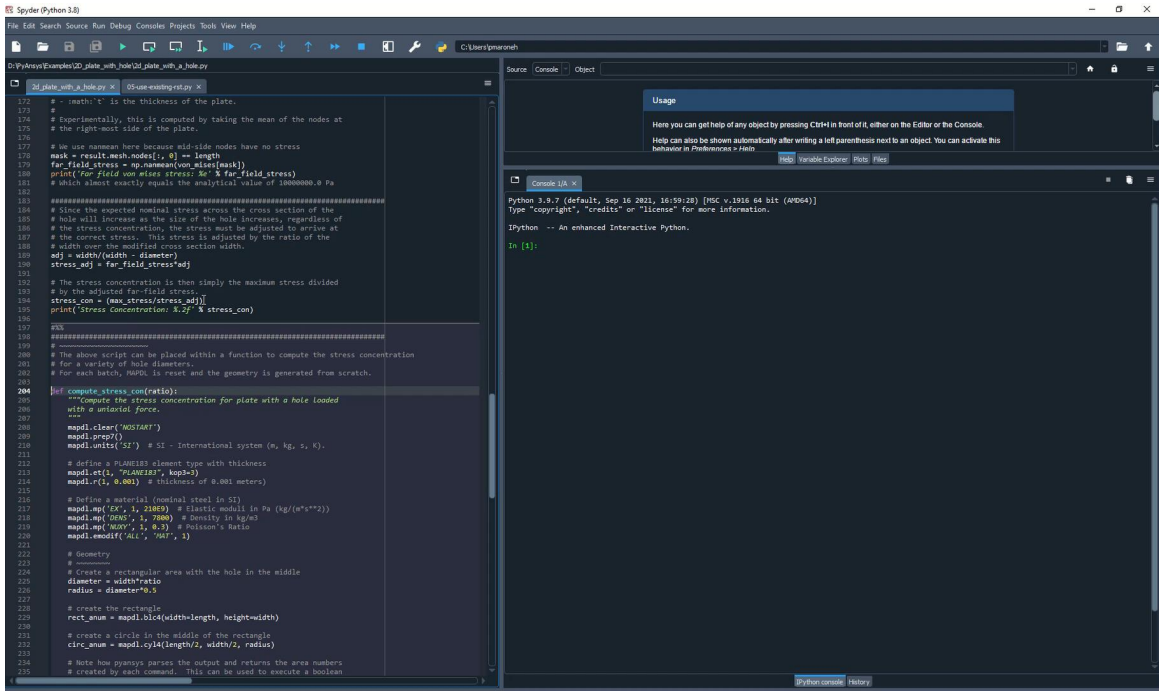
An IDE can be used (PyCharm, Spyder, …)



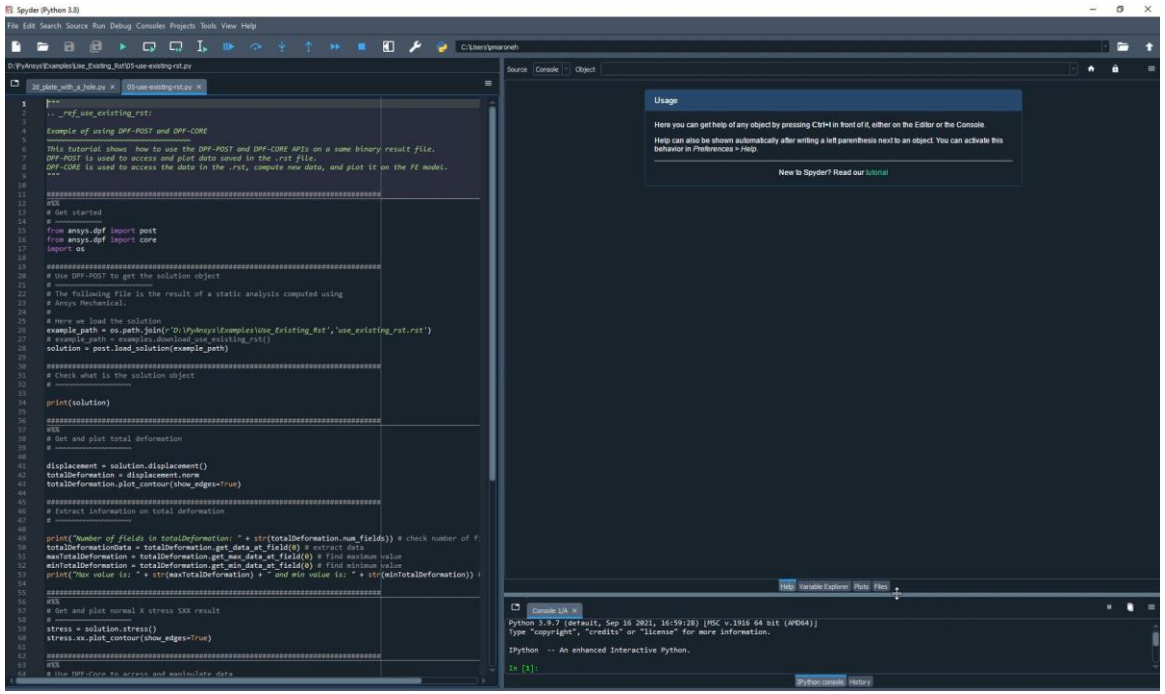
User can also develop apps by coding both the front end and the back end

What can PyAnsys do for me?

Call Ansys products from external Python environment



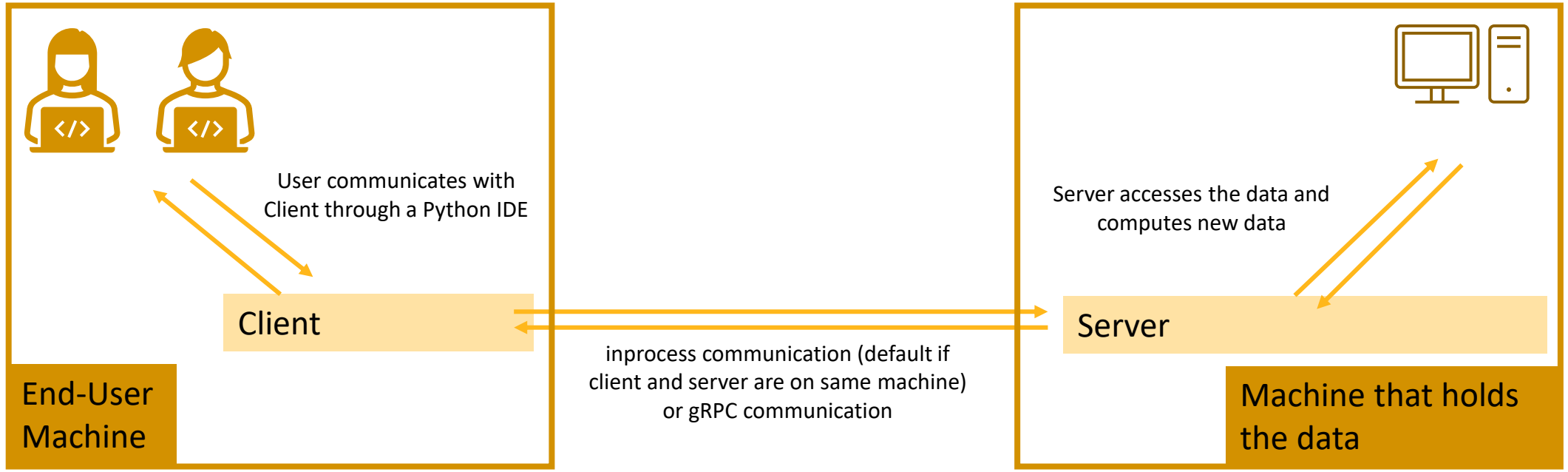
Complete FE setup, run and solving with PyMAPDL



Postprocess a binary result file and create custom results with PyDPF

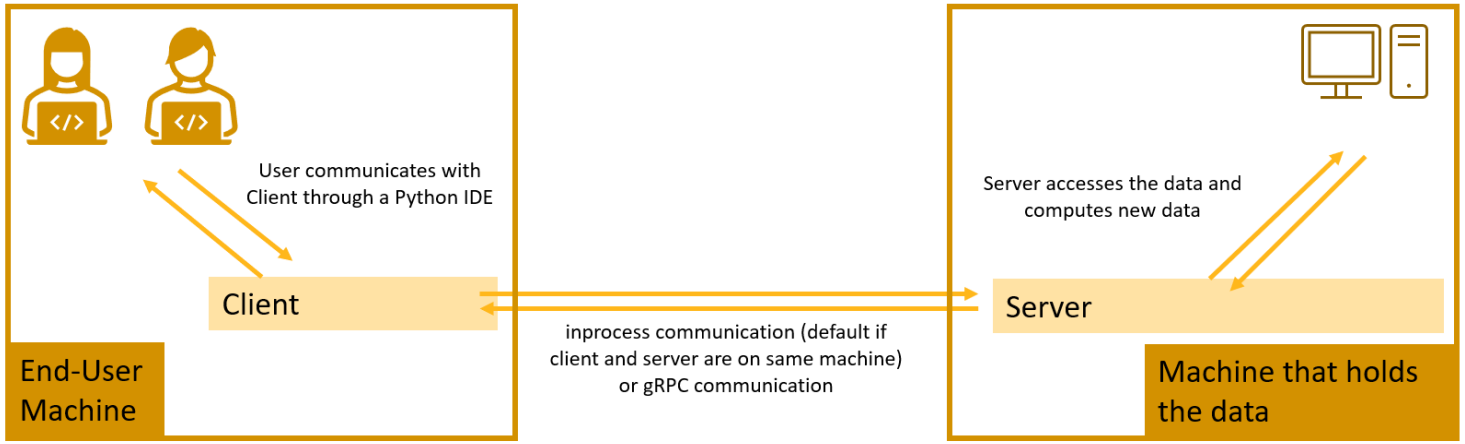
Client/Server architecture

- PyAnsys project offers several packages that are usually based on a Client/Server architecture. Client and Server can be on the same machine, or on separate/remote machines. Below is a typical configuration:

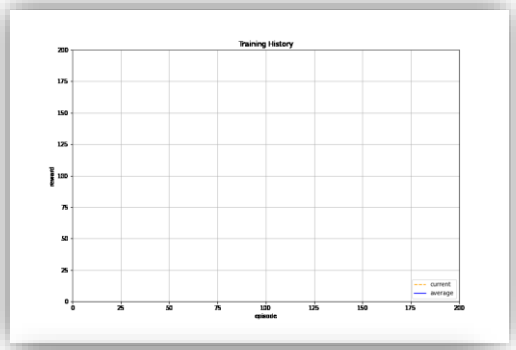


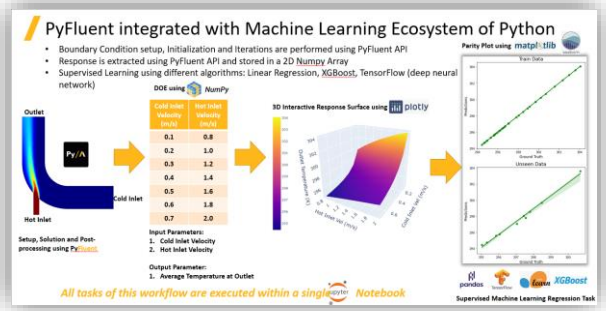
- Where and how to find the files?
  - The Client part comes is provided through Python packages (pip install)
  - The Server part is shipped with the standard Ansys installation

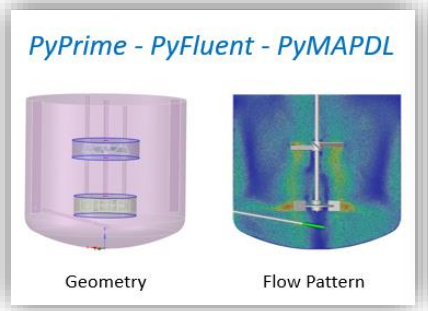
Client/Server architecture

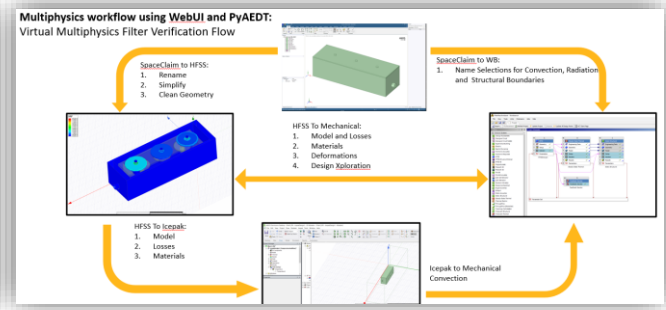


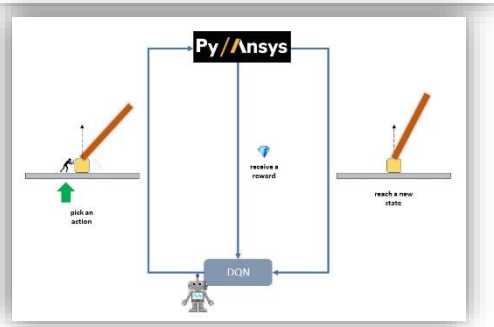
Key workflows where customers are using PyAnsys











Machine Learning Problems

HPC Deployment

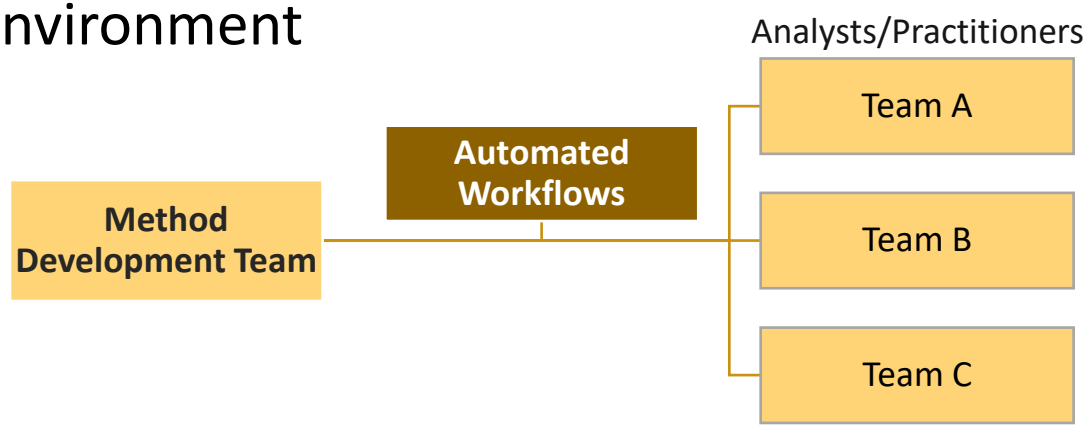
py[Ansys]

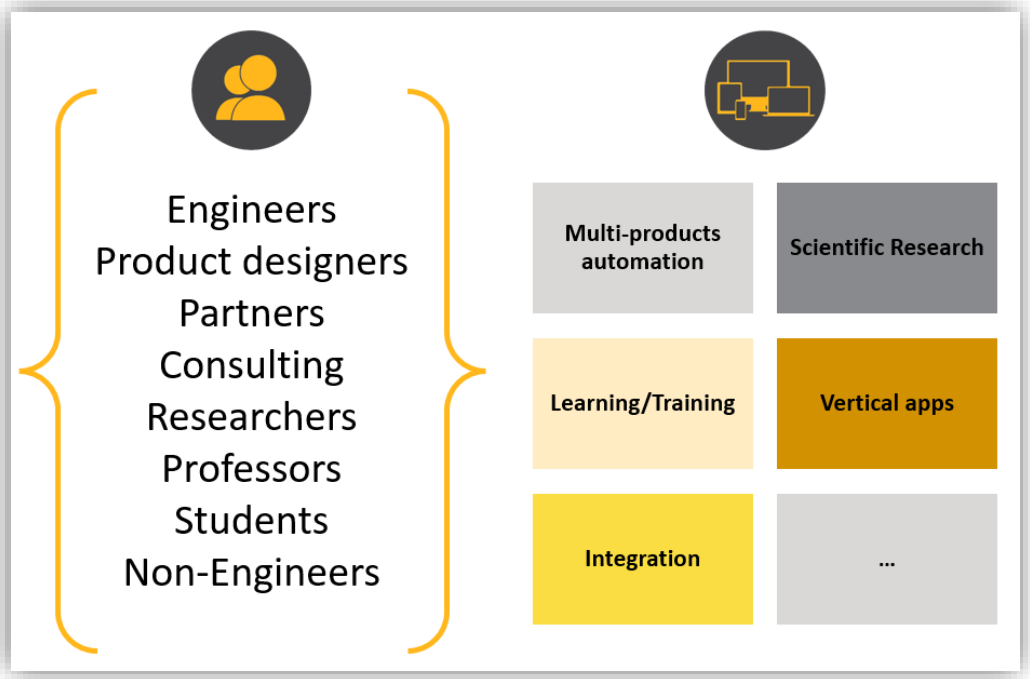
Multi-Physics workflow Automation

Web Applications

Target Personas

- Anyone who wants to **automate** or **integrate** their simulation workflows from outside an Ansys environment





- Integration with scientific computing
  - Computer Vision
  - Artificial Intelligence
  - Machine Learning
  - Data processing and visualization
  - Optimization

How does PyAnsys Makes Difference?

- Allows Control of Ansys tool from external script
- Enables use of Cpython libraries
- Clean, Concise and Maintainable
- Open Source and Well-Documented
- Favorable for AI Pair Programming Tools
- Easy to Deploy and Scalable Workflows
- Enables easy integration with AI/ML Framework
- …Many more!!

How can I install PyAnsys?

How to install PyAnsys?

Follow those three steps:

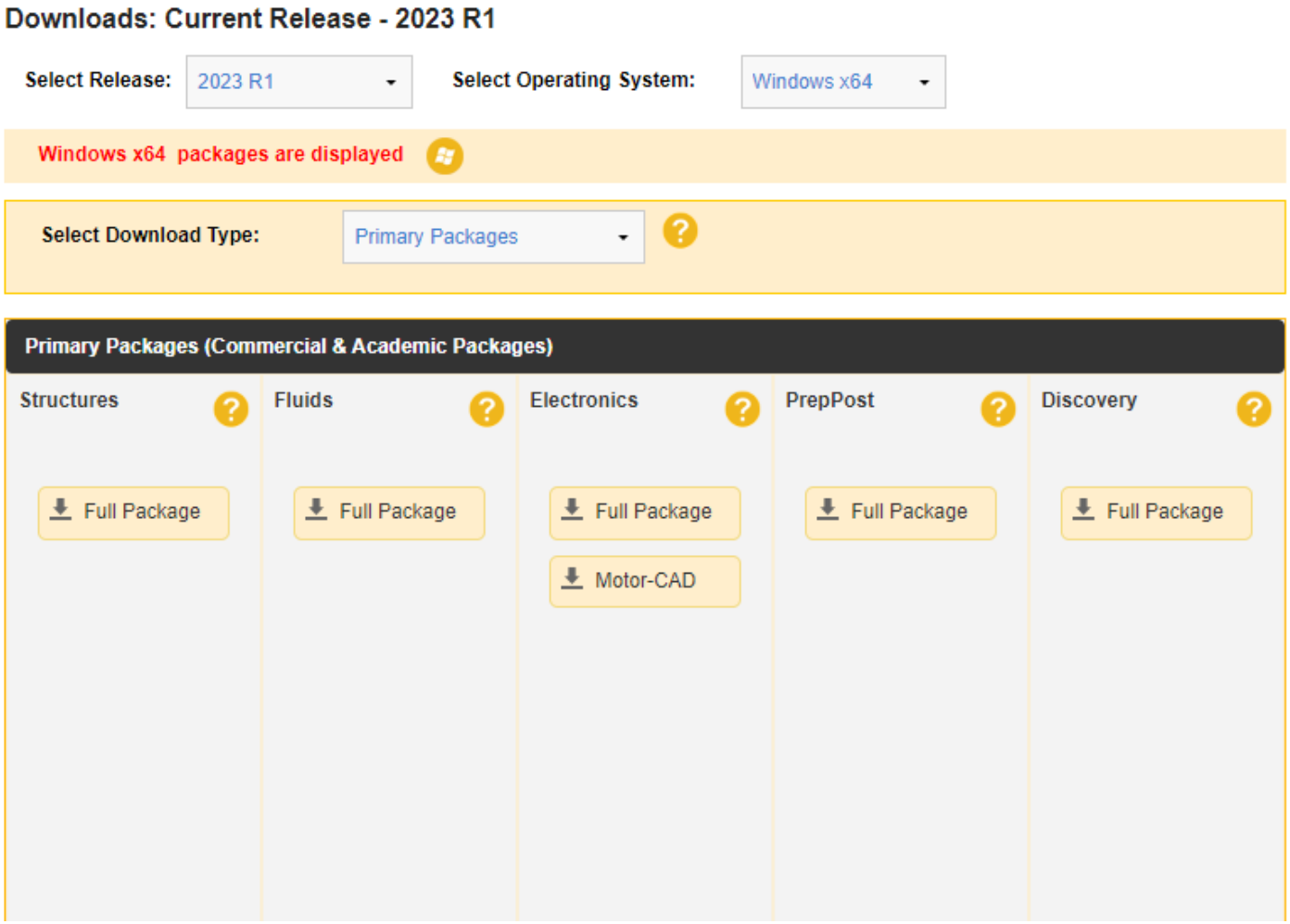
1. Install Ansys Products from standard installation (Mechanical, Fluent, etc.)
2. Install Python and the IDE (code editor) of your choice
3. Install the PyAnsys packages

More info on each step in the next slides!

Alternative to step 2 and 3: use the Ansys Python Manager

How to install it? Step 1: Install Ansys Products

- Download Ansys Products from the Customer Portal
- Using the latest release is recommended to get latest features
- Supported versions:
  - PyMAPDL: Ansys 17.0 and onwards
  - PyDPF: Ansys 2021R1 and onwards



How to install it? Step 2: Install Python and an IDE

- Python version 3.8 or newer is required. Installation from python.org is recommended.
- Use the IDE you are most comfortable with. Some suggestions:
  - Spyder: Free and open-source scientific environment
  - PyCharm: Offers a free community version and full-fledged professional one
  - Visual Studio Code: Free open-source code editor
  - JupyterLab: Web-based interactive development environment

How to install it? Step 3: Install the PyAnsys packages

- Creating and using a virtual environment (venv) is recommended whenever working on different Python based projects. This allows the dependencies of each project to be isolated from each other.
- Python packages are usually installed through “pip” by opening a command prompt and typing “pip install package_name”. PyAnsys packages follow the same procedure:

```
# To install all PyAnsys libraries:
pip install pyansys
```

```
# To install each library individually:
pip install ansys-mapdl-core
pip install ansys-dpf-core
pip install ansys-dpf-post
pip install ansys-dpf-composites
pip install ansys-meshing-prime[all]
pip install ansys-mechanical-core
```

Note: It is also possible to download the wheels from Github.

Ansys Python Manager

- Free, open-source tool
- Seamlessly install Python, PyAnsys packages, and common Python libraries
- Facilitates creation and management of Python virtual environments
- Useful links:
  - GitHub repository
  - Executable
  - Documentation: https://installer.docs.pyansys.com/
  - If you encounter any bugs or have any suggestions for enhancements, log an issue directly in the GitHub repository.

How to launch PyAnsys?

- From a Python IDE, import the package you need and launch it

```
>>> from ansys.mapdl.core import launch_mapdl
>>> mapdl = launch_mapdl()
>>> print(mapdl)

Product:             ANSYS Mechanical Enterprise
MAPDL Version:       RELEASE  2021 R1           BUILD 21.0
PyMAPDL Version:     Version: 0.57.0
```

PyMAPDL

```
>>> from ansys.dpf.core import Model
>>> from ansys.dpf.core import examples
>>> model = Model(examples.find_simple_bar())
>>> print(model)
```

PyDPF-Core

```
import ansys.meshing.prime as prime

with prime.launch_prime() as prime_client:
    model = prime_client.model
```

PyPrimeMesh

```
>>> from ansys.dpf import post
>>> from ansys.dpf.post import examples
>>> simulation = post.load_simulation(examples.download_crankshaft())
>>> displacement = simulation.displacement()
>>> print(displacement)
```

PyDPF-Post

```
from ansys.dpf.composites.composite_model import CompositeModel, CompositeScope
from ansys.dpf.composites.constants import FailureOutput
from ansys.dpf.composites.example_helper import get_continuous_fiber_example_files
from ansys.dpf.composites.failure_criteria import (
    CombinedFailureCriterion,
    CoreFailureCriterion,
    MaxStrainCriterion,
    MaxStressCriterion,
    VonMisesCriterion,
)
from ansys.dpf.composites.server_helpers import connect_to_or_start_server

server = connect_to_or_start_server()
composite_files_on_server = get_continuous_fiber_example_files(server, "shell")
```

PyDPF-Composites

```
import os
from ansys.mechanical.core import launch_mechanical

mechanical = launch_mechanical()
```

```
from ansys.mechanical.core import App

app = App()
ns = app.DataModel.Project.Model.AddNamedSelection()
```

PyMechanical (remote session and embedded instance)

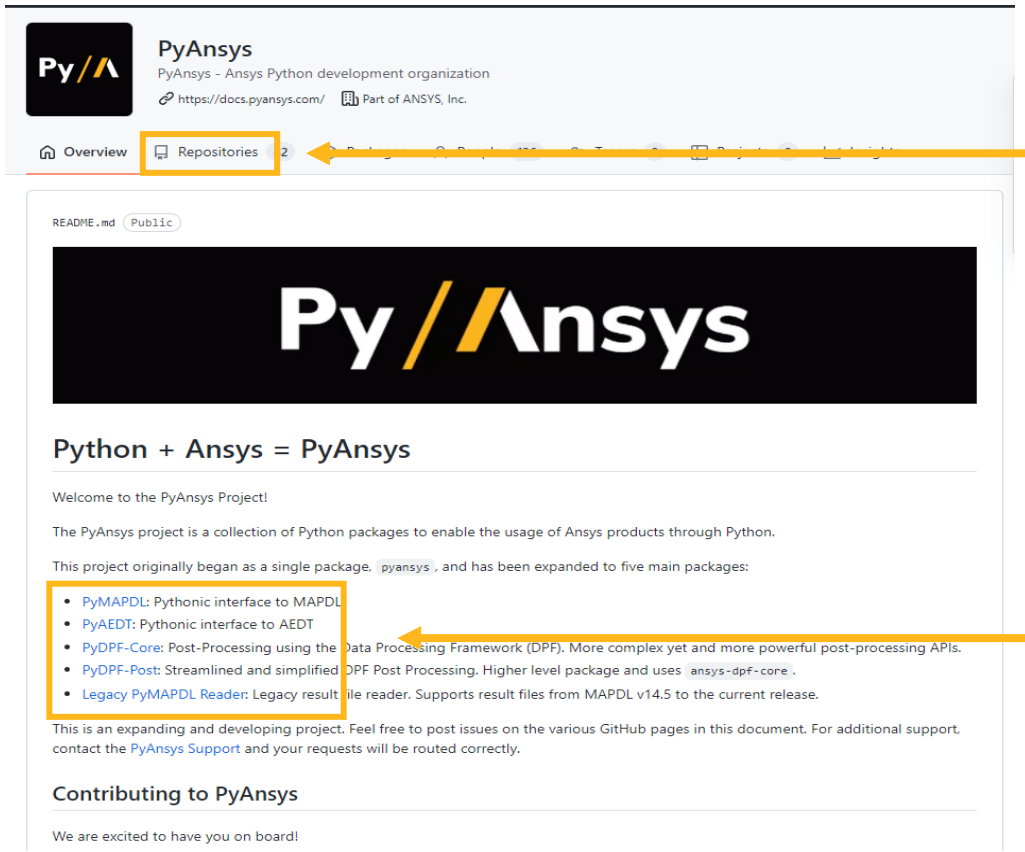
Where are the source code and documentation?
How do I report a bug?
Where can I find help?

Ansys

Source code and Documentation

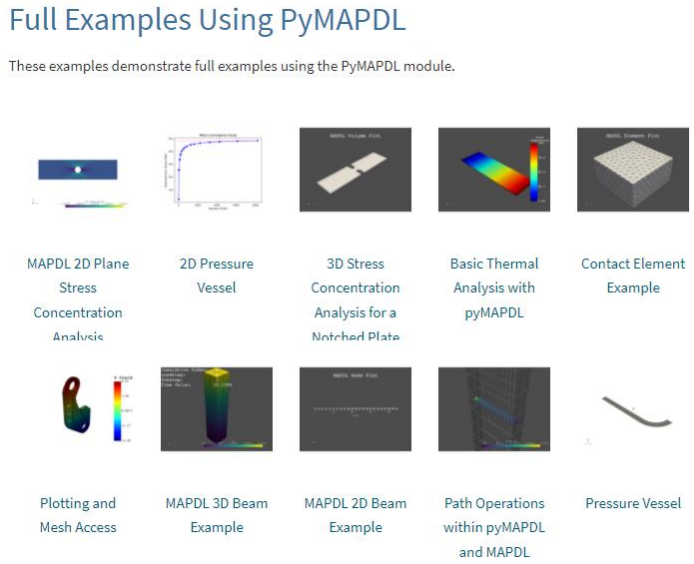
PyAnsys is an open-source project. The source code and documentation are provided on Github.

https://github.com/ansys



Source Code

Documentation

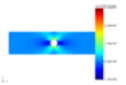


Each package has documentation and many examples

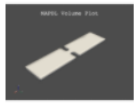
Many examples available

Full Examples Using PyMAPDL

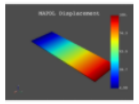
These examples demonstrate full examples using the PyMAPDL module.



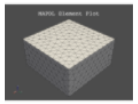
MAPDL 2D Plane Stress Concentration Analysis



3D Stress Concentration Analysis for a Notched Plate

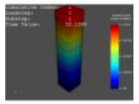


Basic Thermal Analysis with pyMAPDL



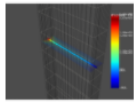
Contact Element Example

Plotting and Mesh Access

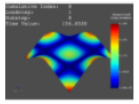


MAPDL 3D Beam Example

MAPDL 2D Beam Example



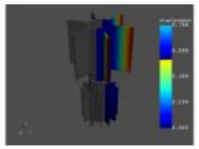
Path Operations within pyMAPDL and MAPDL



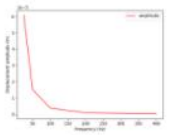
PyVista Mesh Integration

Advanced and Miscellaneous Examples

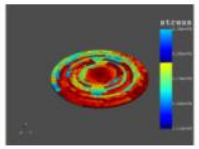
These demos show advanced use cases demonstrating high level of workflow customization



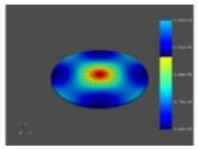
Multi-stage Cyclic Symmetry Use Advanced Customization



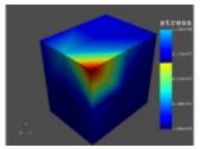
Solve Harmonic Problem (with damping) Using Matrix Inverse



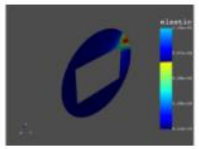
Average Elemental Stress on a given volume



Exchange Data Between Servers



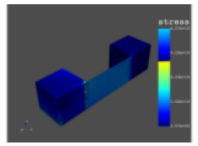
Extrapolation Method for stress result of 3D-element



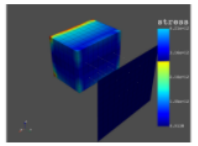
Extrapolation Method for strain result of 2D-element

DPF-Post Analysis Types

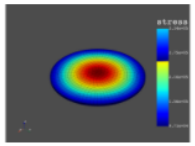
Here are a series of analysis type focused examples with `ansys-dpf-post` module.



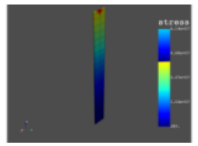
Static Analysis



ANSYS DPF-Post: Modal Analysis



ANSYS DPF-Post: Harmonic Analysis



ANSYS DPF-Post: Transient Anaysis

New examples are being added constantly:

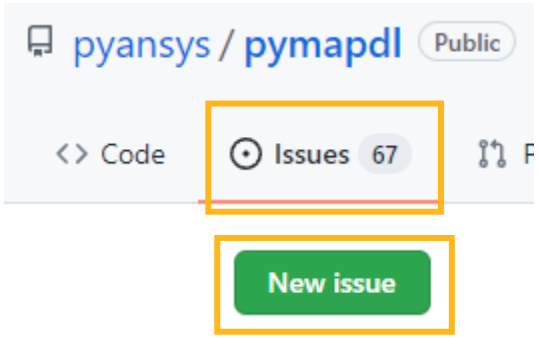
- https://mapdldocs.pyansys.com/
- https://dpfdocs.pyansys.com/
- https://postdocs.pyansys.com/

Documentation Demo

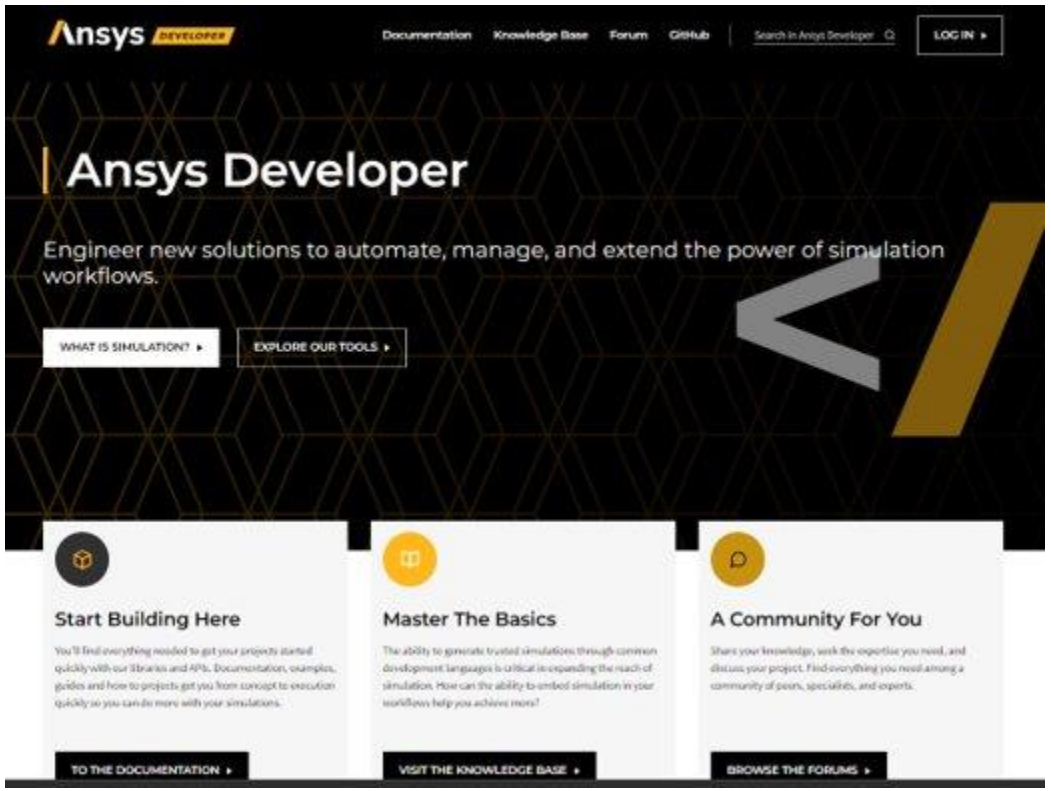
Report a bug, Suggest an enhancement, Want to contribute?

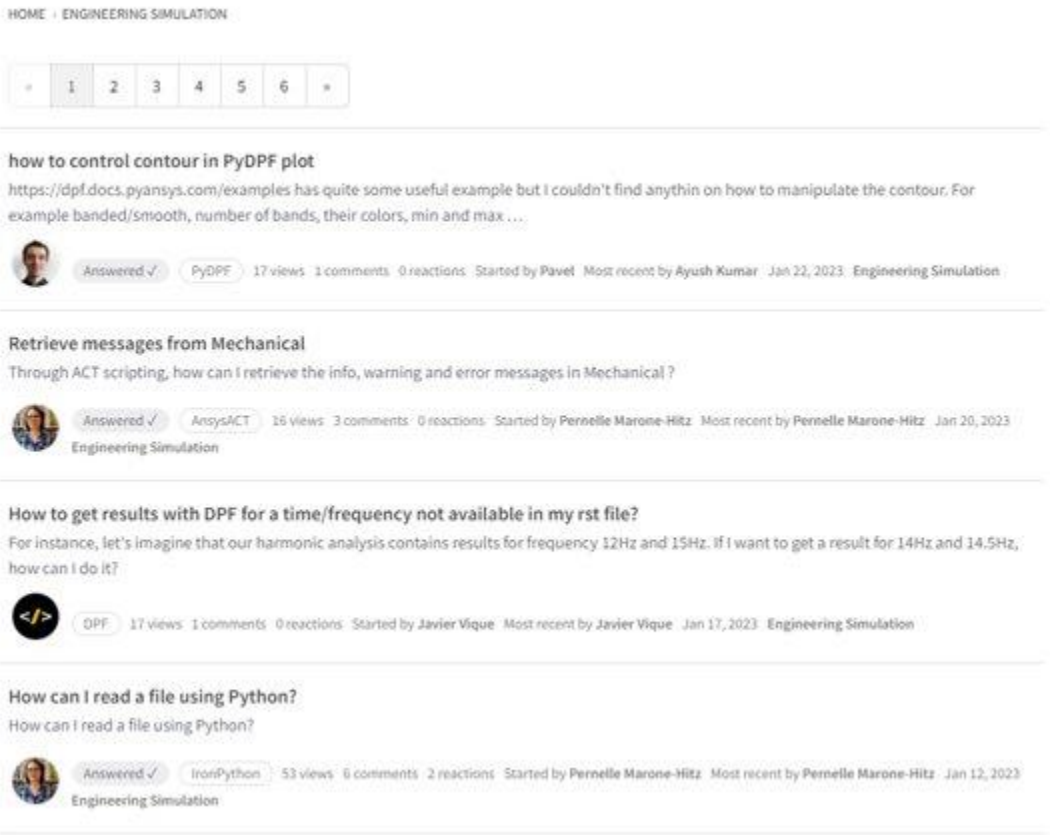
This should be done through Github: open an issue from the Github page of each package.

https://github.com/ansys



Need help? Use the Ansys Developer Portal and Developer Forum





https://developer.ansys.com/

https://discuss.ansys.com/

Training documents

Some PyAnsys courses or other useful courses are available on the Ansys Innovation Space and the Ansys Learning Hub

STRUCTURES

LEARN SIMULATION

Getting Started with Ansys PyMAPDL

LEARN MORE ⟶

STRUCTURES

LEARN SIMULATION

Developing WebApps for Modeling and Simulation Using PyAnsys

LEARN MORE ⟶

PYTHON

LEARN SIMULATION

Intro to Python

LEARN MORE ⟶

STRUCTURES

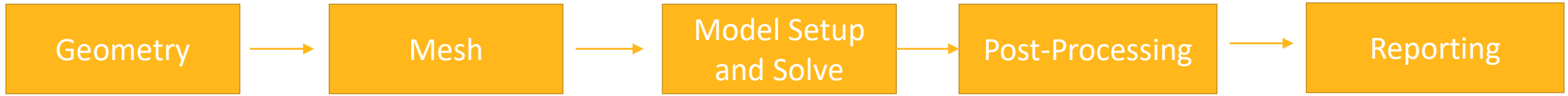
LEARN SIMULATION

Intro to Ansys APDL Scripting

LEARN MORE ⟶

Overview of Modules

A Typical Structural Simulation Workflow



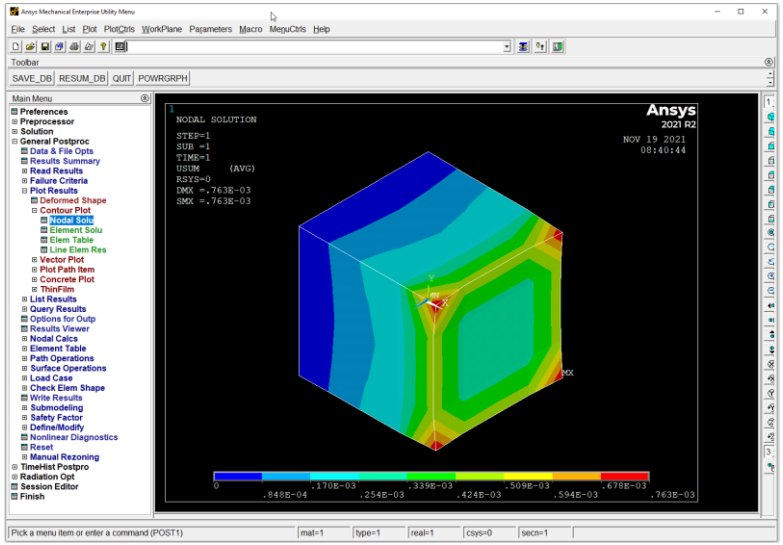
Geometry	Mesh	Model Setup and Solve	Post-Processing	Reporting
SpaceClaim/Discovery	Mechanical	Mechanical	Mechanical	Nexus
PyAnsysGeometry	PyMechanical	PyMechanical	PyMechanical	PyDynamicReporting
	Prime	MAPDL	DPF	
	PyPrimeMesh	PyMAPDL	PyDPF	
		LS-DYNA		
		PyDYNA		

PyMAPDL

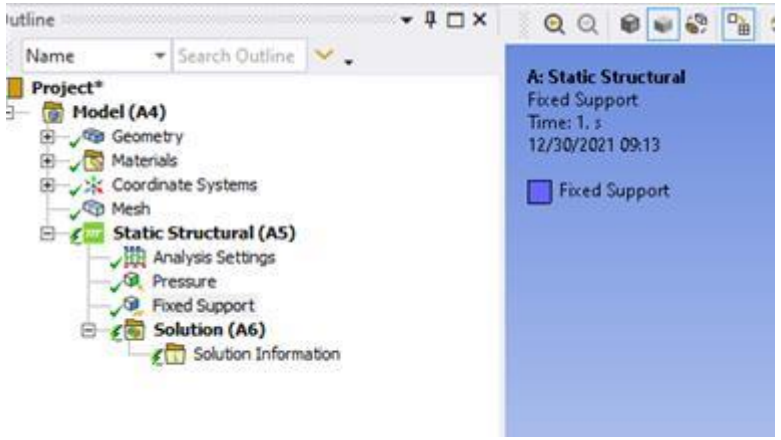
Ansys

MAPDL = Mechanical Ansys Parametric Design Language

Scripting language used for the Ansys Mechanical solver



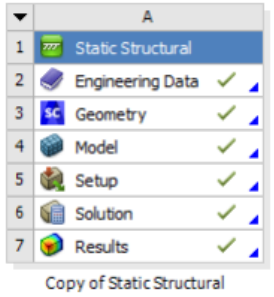
Ansys Classic



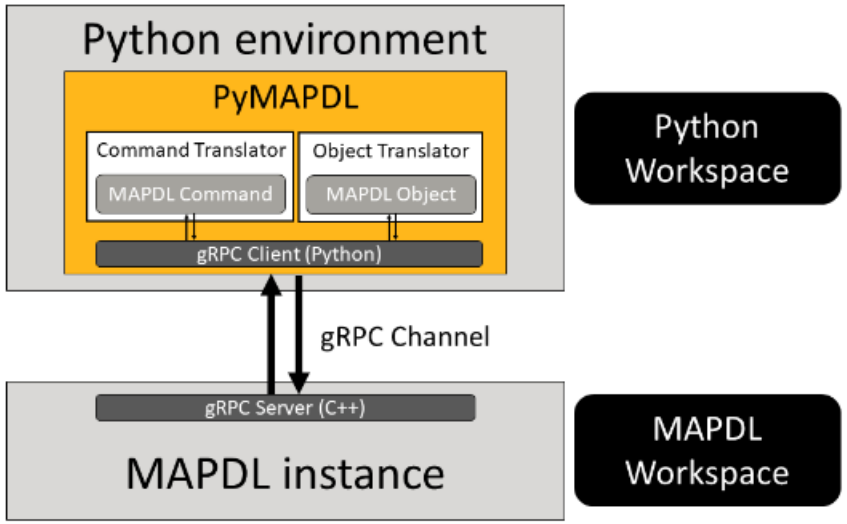
Mechanical

```
/PREP7
x = 1
y = x
z = x
f = 1e9
MP , EX , 1 , 2.1E+14
MP , PRXY , 1 , 0.3
BLOCK , 0 , x , 0 , y , 0 , z
ET , 1 , 186
ESIZE , x/4
VMESH, ALL
NSEL , S , LOC , X , 0 ,
D , ALL , ALL
NSEL , S , LOC , X , x
F,ALL,FX,f
ALLSEL , ALL
/SOL
SOLVE
FINISH
/POST1
PLDISP,2
```

Both rely on MAPDL

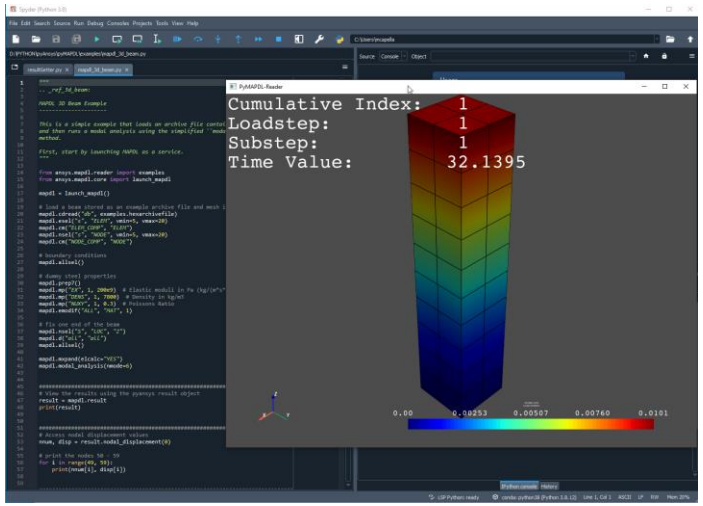


From MAPDL to PyMAPDL



PyMAPDL architecture diagram

- ❑ Use MAPDL in a Python environment
- ❑ Interface of users' choice
- ❑ Python ease of use:
  - *do/*enddo -> for i in data
  - *get -> mapdl.parameters["MY_ARRAY"]
- ❑ PyMAPDL can be combined with any other python packages



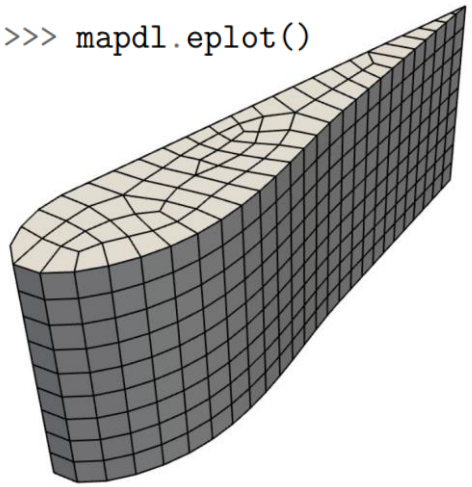
PyMAPDL

- With PyMAPDL, it's possible to:
  - Create geometry, mesh, model setup
  - Do interactive plotting
  - Post-process
  - Translate MAPDL scripts to PyMAPDL script

PyAnsys Code Example: Meshing

```
>>> mapdl.et(1, 'SOLID186')
>>> mapdl.vsweep('ALL')
>>> mapdl.esize(0.1)
>>> mapdl.mesh
 ANSYS Mesh
   Number of Nodes:              7217
   Number of Elements:           2080
   Number of Element Types:      2
   Number of Node Components:    0
```

```
>>> mapdl.eplot()
```



- Either use standard APDL commands through "Run" command, or call MAPDL Pythonically:

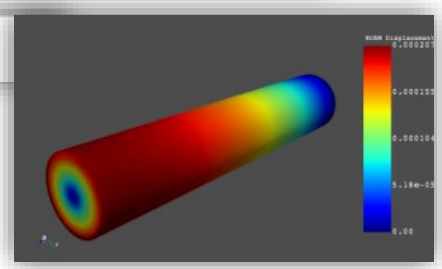
```
mapdl.run('/PREP7')
mapdl.run('K, 1, 0, 0, 0')
mapdl.run('K, 2, 1, 0, 0')
mapdl.run('K, 3, 1, 1, 0')
mapdl.run('K, 4, 0, 1, 0')
mapdl.run('L, 1, 2')
mapdl.run('L, 2, 3')
mapdl.run('L, 3, 4')
mapdl.run('L, 4, 1')
mapdl.run('AL, 1, 2, 3, 4')
```

```
mapdl.prep7()
mapdl.k(1, 0, 0, 0)
mapdl.k(2, 1, 0, 0)
mapdl.k(3, 1, 1, 0)
mapdl.k(4, 0, 1, 0)
mapdl.l(1, 2)
mapdl.l(2, 3)
mapdl.l(3, 4)
mapdl.l(4, 1)
mapdl.al(1, 2, 3, 4)
```

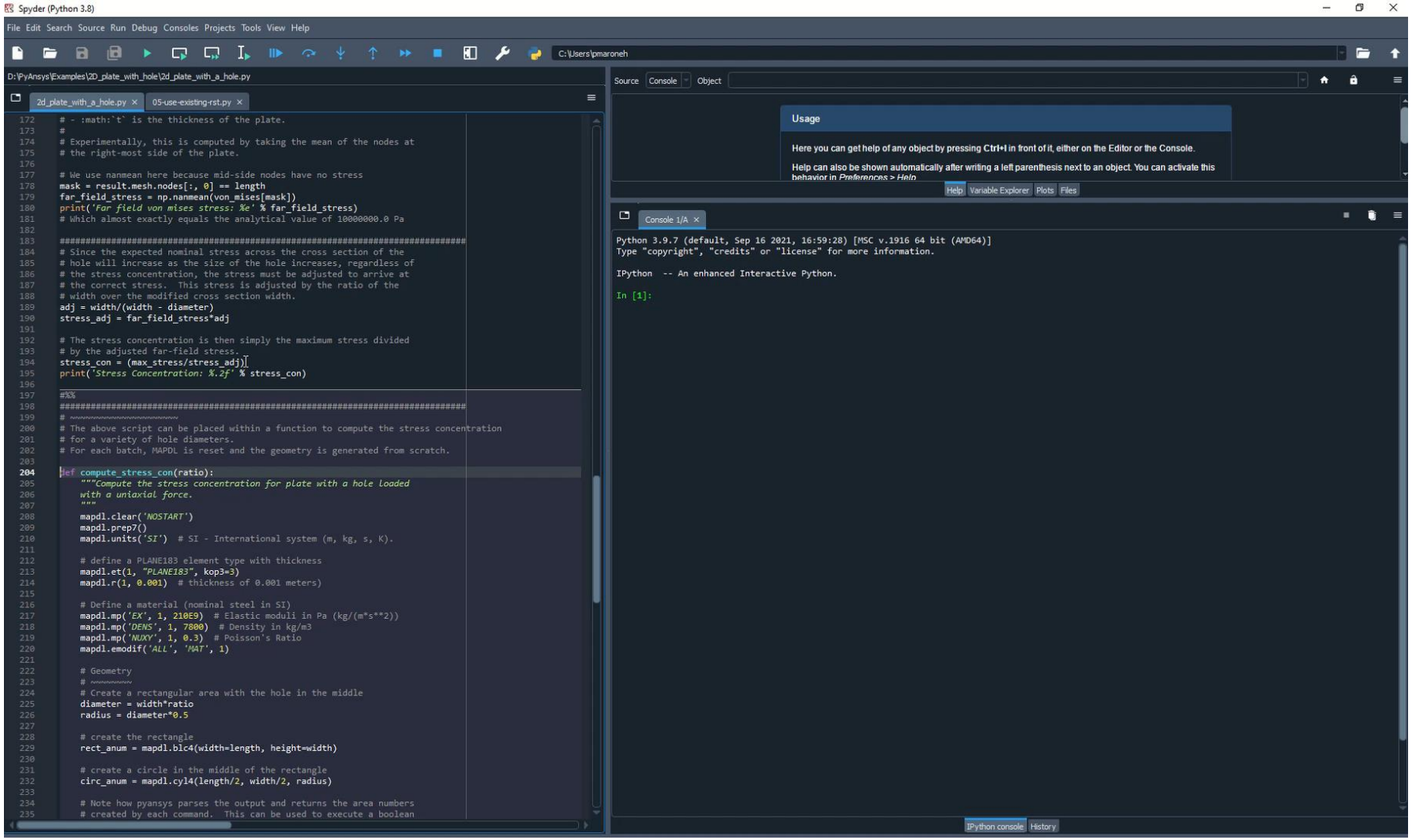
```
POST1:
PRNSOL, U, X

PRINT U    NODAL SOLUTION PER NODE

***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

LOAD STEP=     1  SUBSTEP=     1
 TIME=    1.0000      LOAD CASE=   0

THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM

  NODE       UX
     1  0.10751E-003
     2  0.85914E-004
     3  0.57069E-004
     4  0.13913E-003
     5  0.35621E-004
     6  0.52186E-004
     7  0.30417E-004
```

```
>>> mapdl.set(1, 1)
>>> disp_x = mapdl.post_processing.nodal_displacement('X')
array([1.07512979e-04, 8.59137773e-05, 5.70690047e-05, ...,
       5.70333124e-05, 8.58600402e-05, 1.07445726e-04])
```

```
>>> mapdl.post_processing.plot_nodal_displacement('X')
```



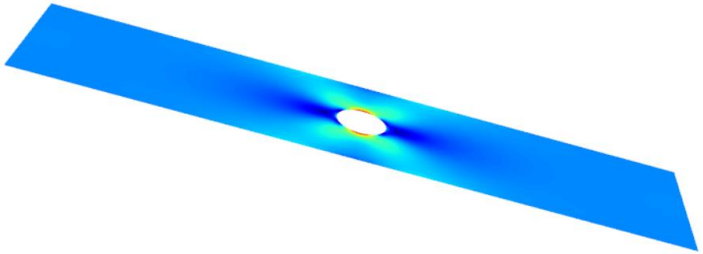
Demo PyMAPDL – evaluate stress concentration factor on a 2D plate with a hole



Demo PyMAPDL – evaluate stress concentration factor on a 2D plate with a hole

```
import matplotlib.pyplot as plt
import numpy as np

from ansys.mapdl.core import launch_mapdl

mapdl = launch_mapdl()
print(mapdl)

#%%
################################################################################
# Element Type and Material Properties
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# This example will use PLANE183 elements as a thin plate can be
# modeled with plane elements provided that KEYOPTION 3 is set to 3
# and a thickness is provided.
#
# This example will use SI units.

mapdl.prep7()
mapdl.units('SI')  # SI - International system (m, kg, s, K).

# define a PLANE183 element type with thickness
mapdl.et(1, "PLANE183", kop3=3)
mapdl.r(1, 0.001)  # thickness of 0.001 meters)

# Define a material (nominal steel in SI)
mapdl.mp('EX', 1, 210E9)  # Elastic moduli in Pa (kg/(m*s**2))
mapdl.mp('DENS', 1, 7800)  # Density in kg/m3
mapdl.mp('NUXY', 1, 0.3)  # Poisson's Ratio

# list currently defined material properties
print(mapdl.mplist())
```

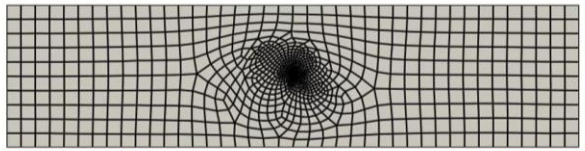
Model definition with PyMAPDL

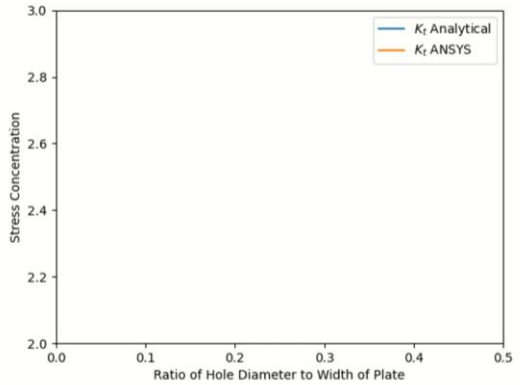


Interactive plotting (mesh, results, etc)

```
Far field von mises stress: 9.999966e+06
Stress Concentration: 2.34
    Ratio  : Stress Concentration (K_t)
    0.0100 :     3.3066
    0.0450 :     2.8606
    0.0800 :     2.7565
    0.1150 :     2.6636
    0.1500 :     2.5838
    0.1850 :     2.5031
    0.2200 :     2.4326
    0.2550 :     2.3653
    0.2900 :     2.3061
    0.3250 :     2.2514
    0.3600 :     2.2017
    0.3950 :     2.1551
    0.4300 :     2.1148
    0.4650 :     2.0777
    0.5000 :     2.0411
```

Batch analysis to determine Stress concentration factor (ratio of maximum stress at hole to the far field stress)



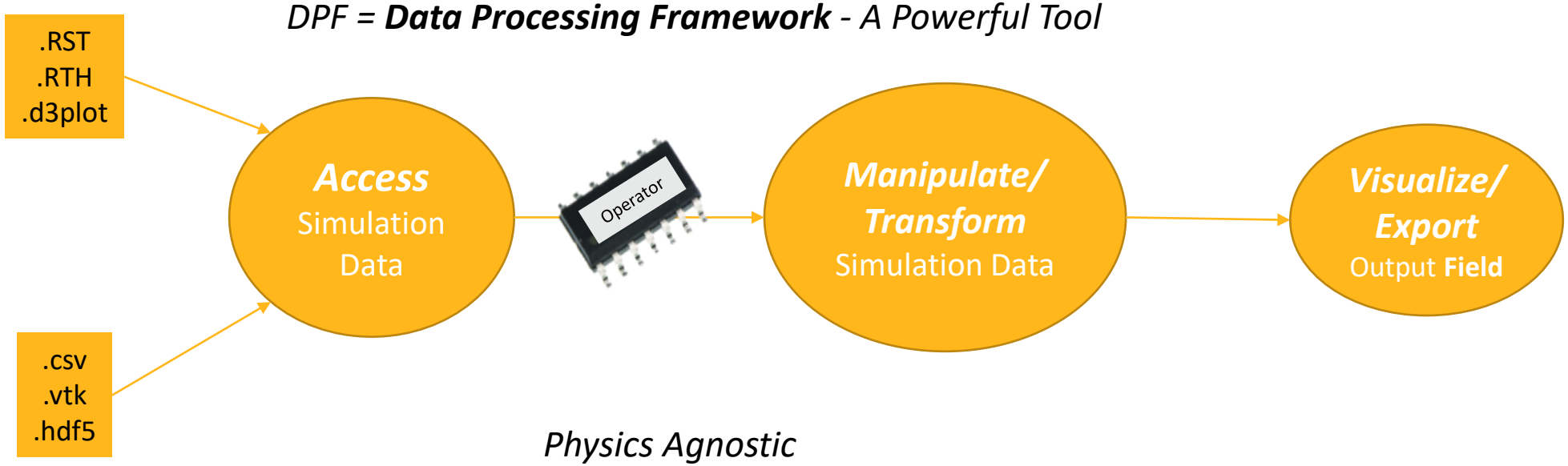


PyDPF

Ansys

What is DPF?

*DPF = **Data Processing Framework** - A Powerful Tool*



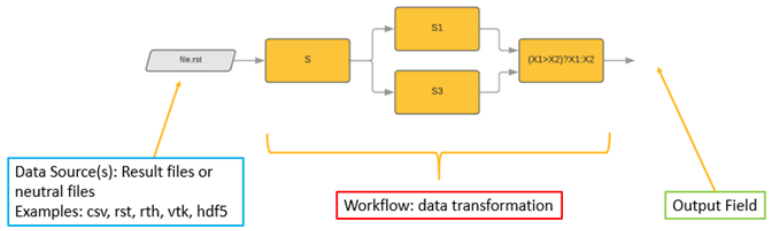
Physics Agnostic
Performance
Remote/Distributed Access
Pythonic
Customizable
Open Source
Dedicated Libraries (eg: numpy, matplotlib)
Plug & Play

DPF in PyAnsys

3 packages based on DPF in PyAnsys

DPF Core

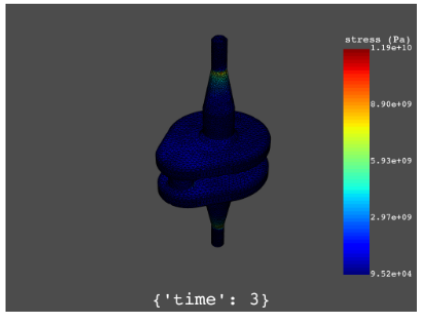
Read and manipulate finite element data with powerful, scalable operators



DPF Post

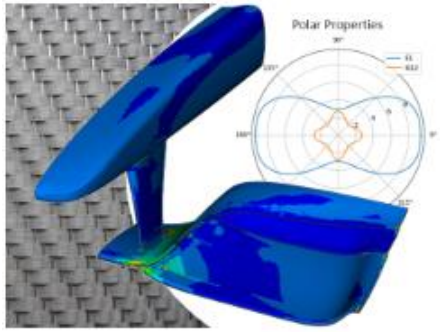
Simplified post-processing API, Uses DPF-Core Behind the scenes

```
>>> stress_eqv = simulation.stress_eqv_von_mises_nodal()
>>> stress_eqv.plot()
```



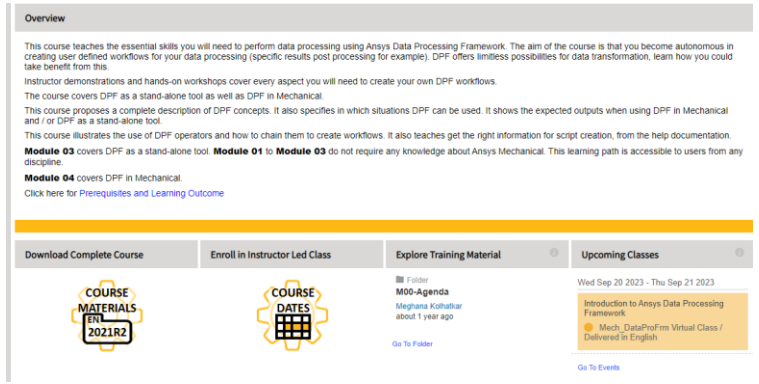
DPF Composites

Additional classes for short fiber and layered composites (shell and solid elements)

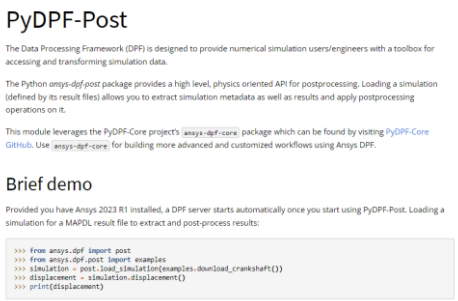


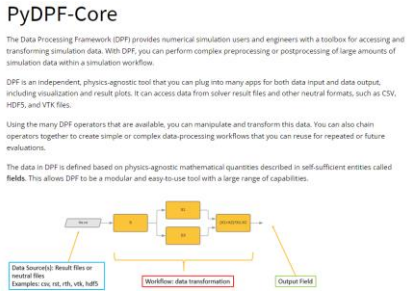
Training material

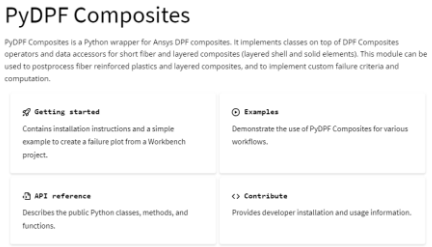
- Ansys Learning Hub:
  - Introduction to Ansys Data Processing Framework



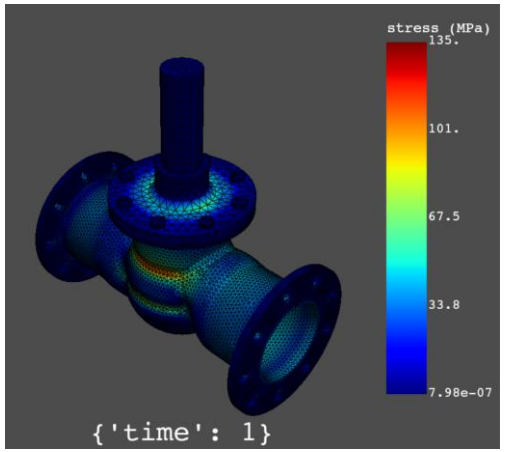
- PyDPF documentation and examples:
  - PyDPF-Post
  - PyDPF-Core
  - PyDPF-Composites

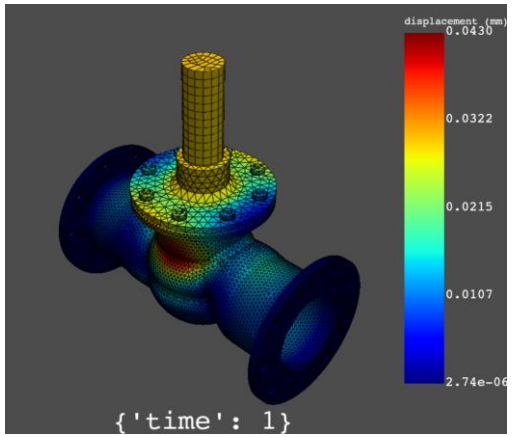






PyDPF-Post – Simplified API for postprocessing

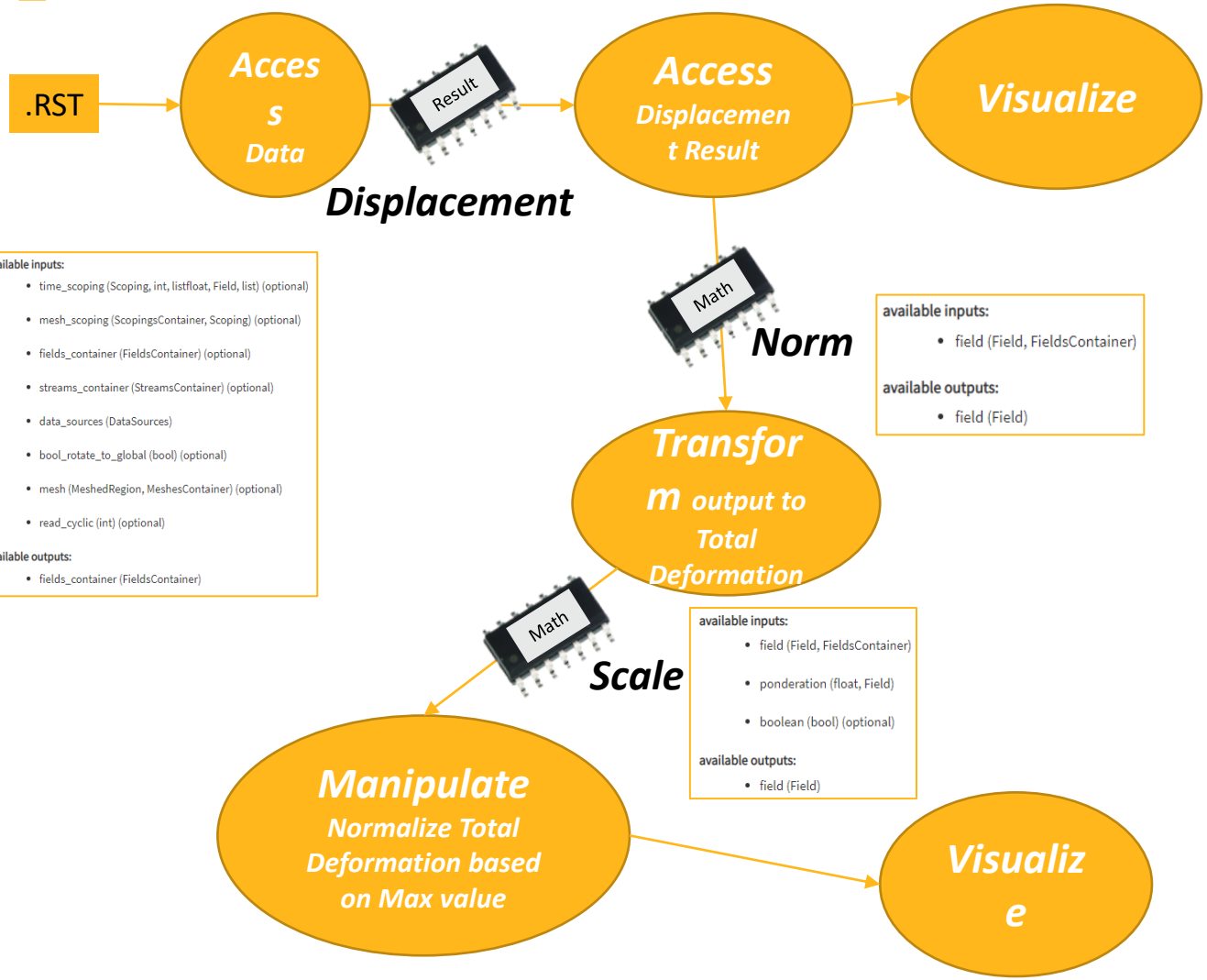




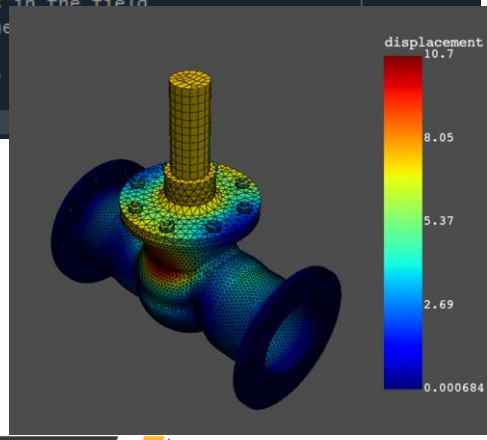

```
Max value is: 0.04295543409307043 and min value is: 2.7364042403964693e-06
```

```python
# Get started
# ~~~~~~~~~~~~~~~
from ansys.dpf import post
import os

# Use DPF-POST to get the solution object
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~
# The following file is the result of a static analysis computed using
# Ansys Mechanical.
#
# Here we load the solution
example_path = os.path.join(r'D:\PyAnsys\PyAnsys Workshop\Examples','valve_result_file.rst')

simulation = post.load_simulation(example_path)

# Check what is the solution object
# ~~~~~~~~~~~~~~~~~~~~~~
print(simulation)

#%%
# Get and plot total deformation
# ~~~~~~~~~~~~~~~~~~~~~~

total_deformation = simulation.displacement(norm = True)
total_deformation.plot()

# Extract information on total deformation
# ~~~~~~~~~~~~~~~~~~~~~~
print(total_deformation)

total_deformation_data = total_deformation.array # extract data
max_total_deformation = total_deformation.array.max() # find maximum value
min_total_deformation = total_deformation.array.min() # find minimum value
print("Max value is: " + str(max_total_deformation) + " and min value is: " + str(min_total_deformation))

#%%
# Get and plot Von Mises stress
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
stress_eqv = simulation.stress_eqv_von_mises_nodal()
stress_eqv.plot()
```

PyDPF-Core

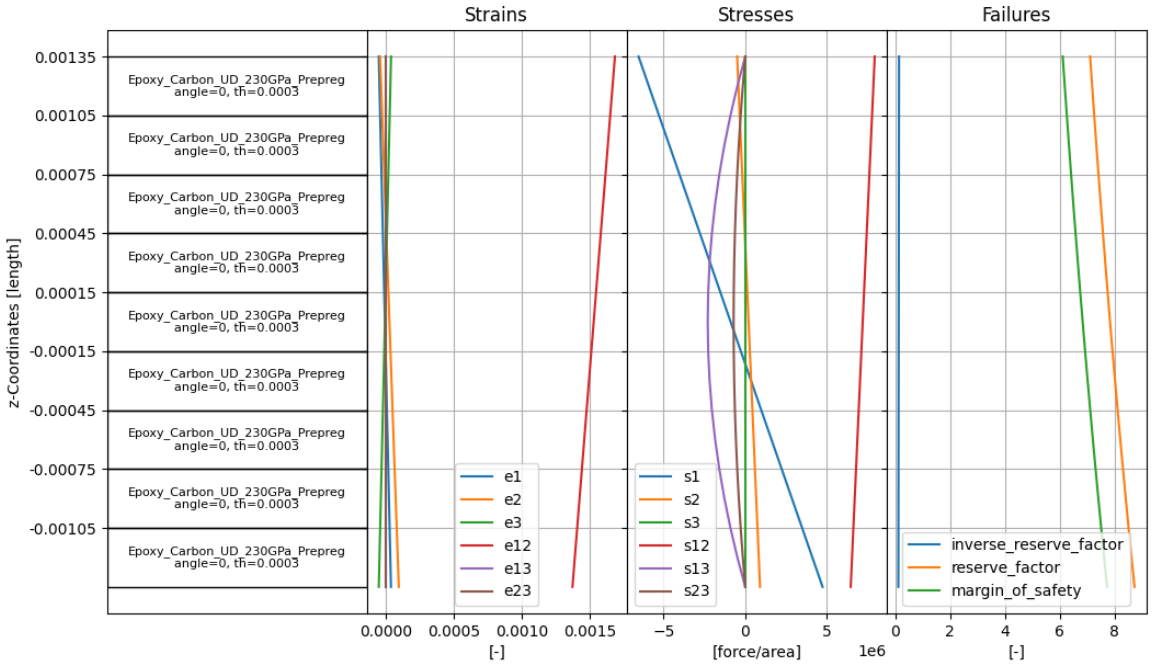


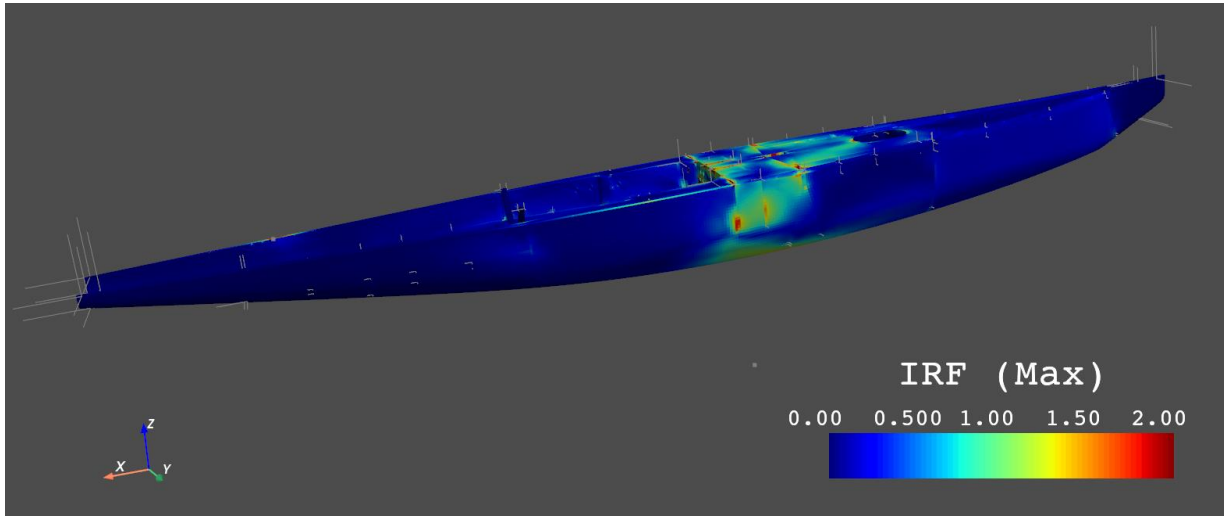

```python
from ansys.dpf import core as dpf
import os

example_path = os.path.join(r'D:\PyAnsys\Examples\Use_Existing_Rst','use_existing_rst.rst')

model = dpf.Model(example_path)
print(model) # print all info contained in .rst

# print info separately
metadata = model.metadata
print(metadata.result_info)
print(metadata.meshed_region)
print(metadata.time_freq_support)

displacement_op = model.results.displacement()
disp_field_container = displacement_op.outputs.fields_container()
print(disp_field_container[0].data)

norm_op = dpf.operators.math.norm()
norm_op.inputs.field.connect(disp_field_container)
total_deformation = norm_op.outputs.field()

scale_op = dpf.Operator('scale') # operator instanciation
scale_op.inputs.field.connect(total_deformation)
scale_value = float(250)
scale_op.inputs.ponderation.connect(scale_value)

my_new_field = scale_op.outputs.field()
print(my_new_field) # Check what information is in the field
print(my_new_field.data) # Check the data in the

model.metadata.meshed_region.plot(my_new_field)
```



PyDPF-Composites

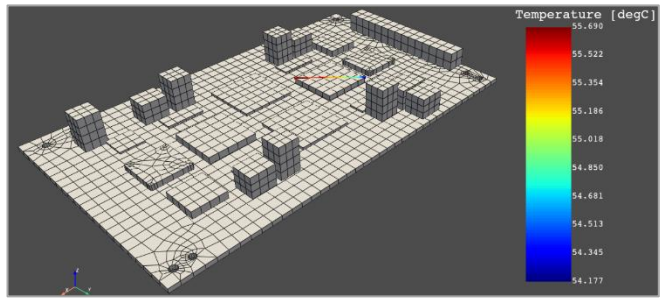
- Python wrapper for post-processing of composite models
- Additional classes for short fiber and layered composites
- Handles shell and solid elements
- Custom failure criteria
- Sampling points

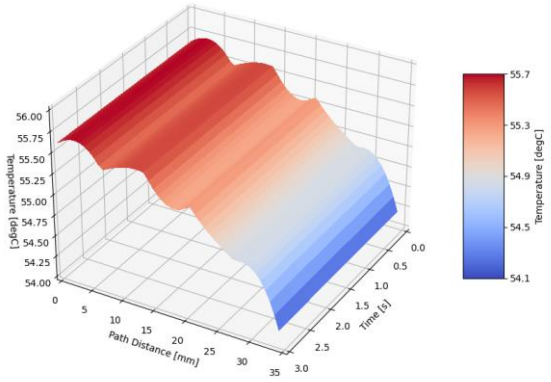




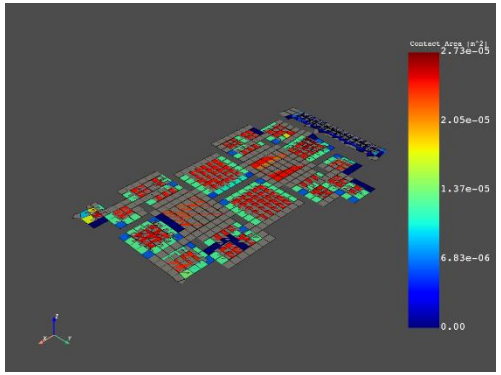
Examples

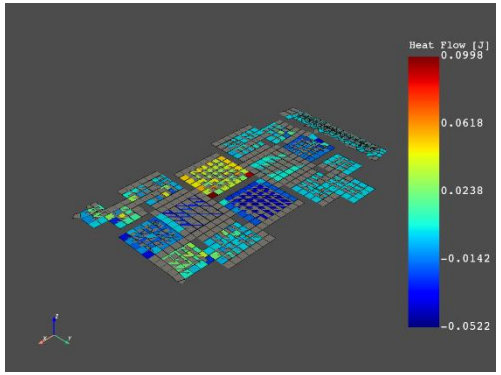
Extracting path and creating 3D Surface plot



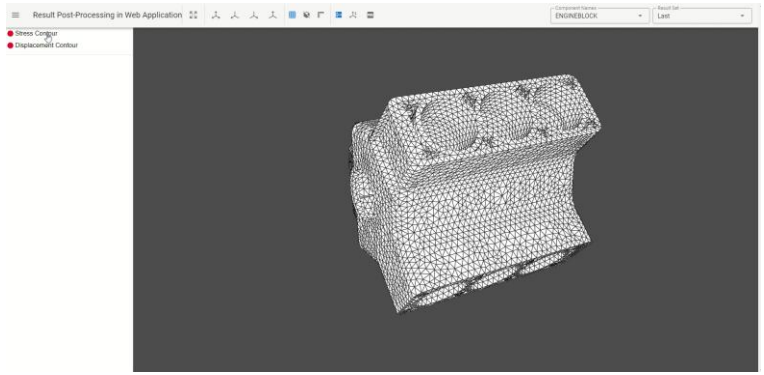


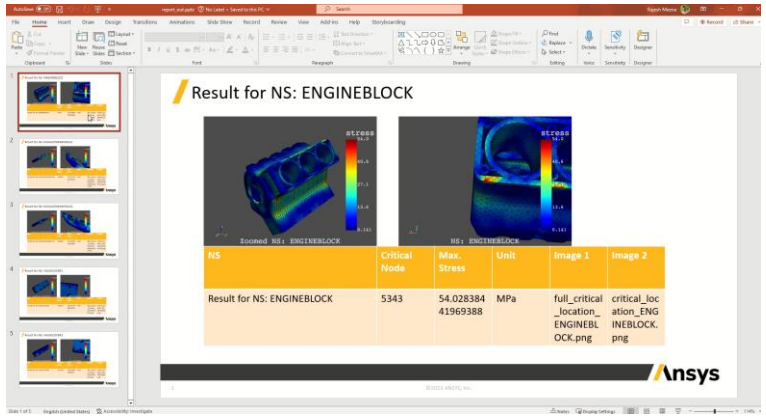
Contact Area and heat flow plotting



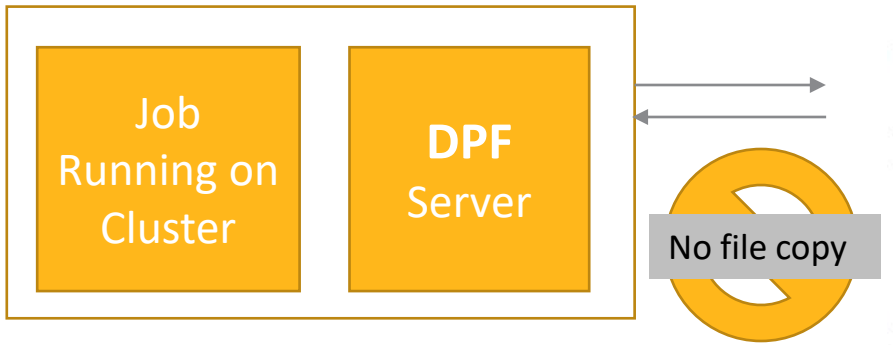


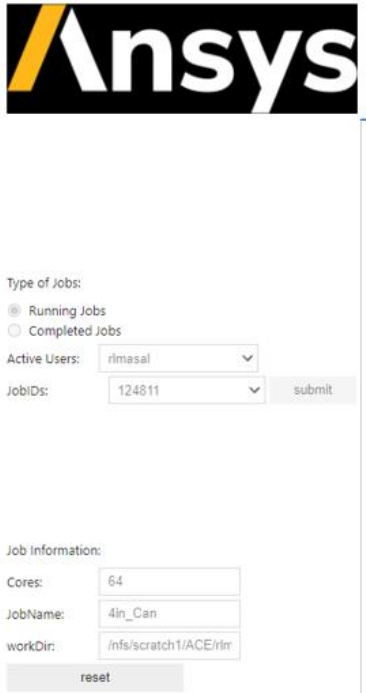
Web App and Automatic Report Generation

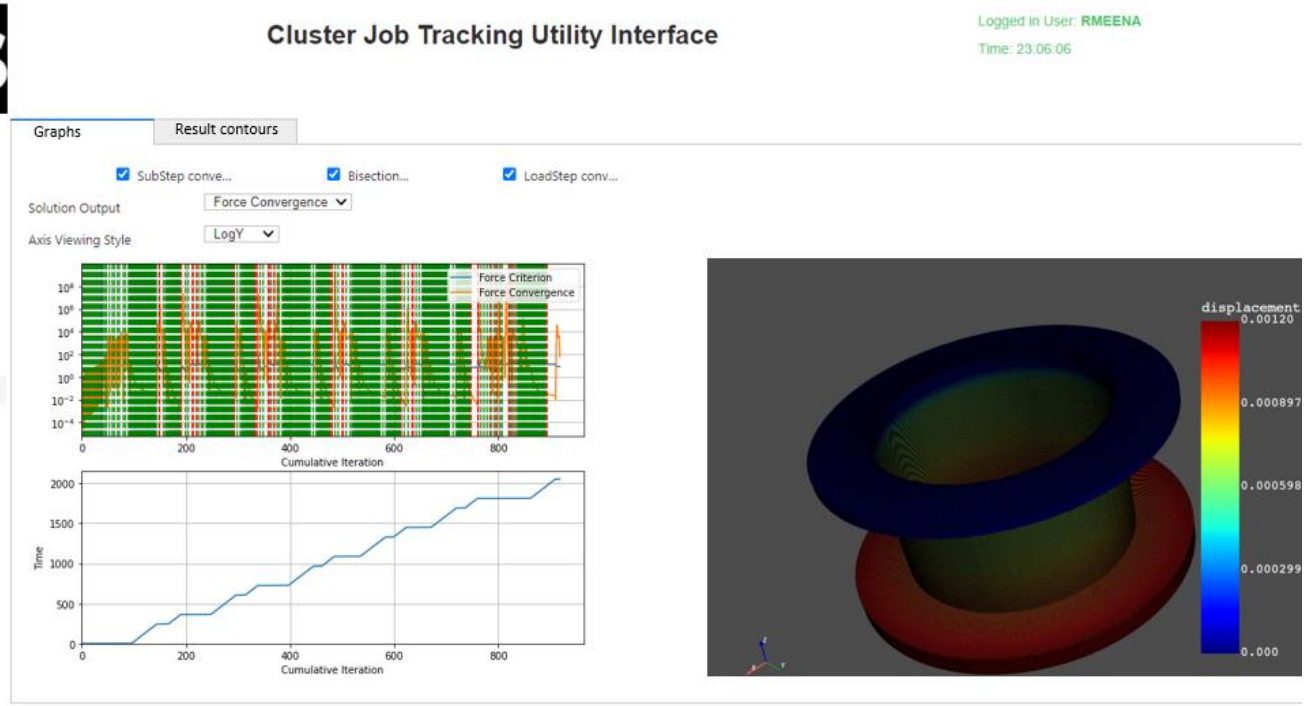




Cluster Job Tracking WebApp using pyAnsys







Tasks on Server/Cluster

Realtime tracking on client side

- ✓ Uses pyAnsys APIs at backend
- ✓ Realtime tracking
  - ✓ Contacts, convergence graphs
  - ✓ Result contours

Possibilities: More diagnostic features; Can be available on smartphone

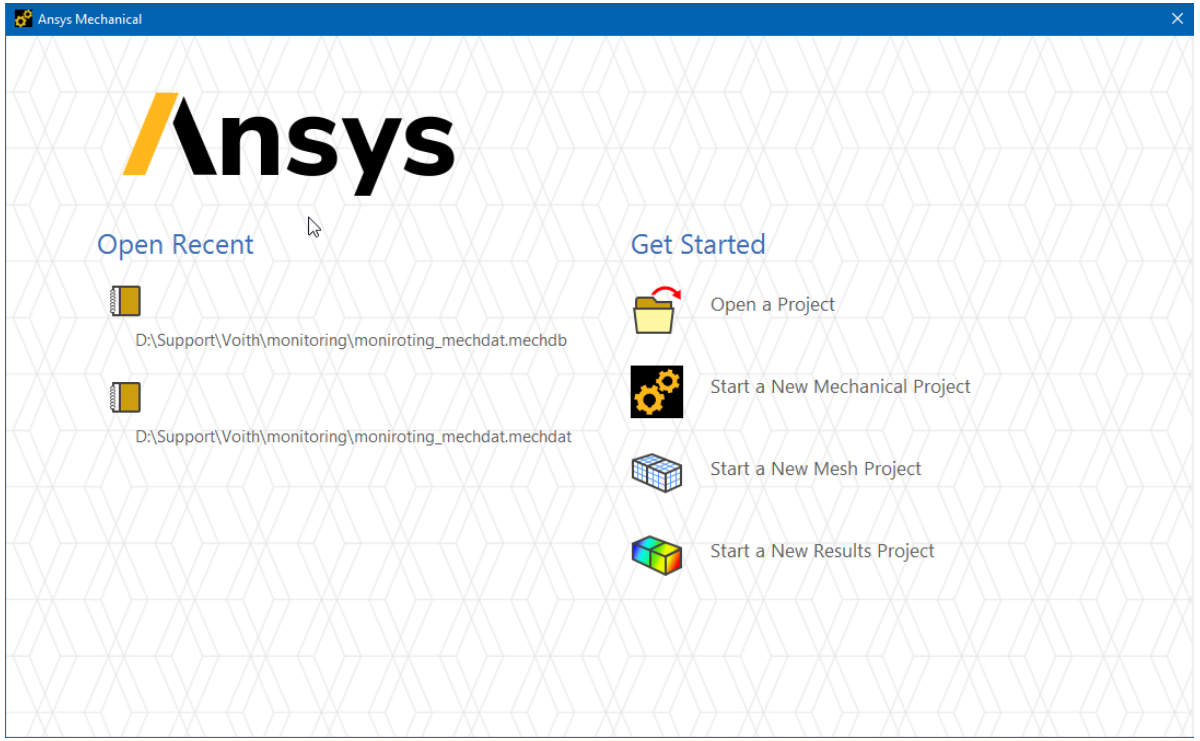
PyMechanical

Ansys

What is Mechanical Standalone?

Standalone Mechanical

- Mechanical without dependencies to the project page: Geometry, External Data, Engineering Data, Data transfer and multiphysics links …
- Allows users to access and automate Mechanical directly without having to script WB

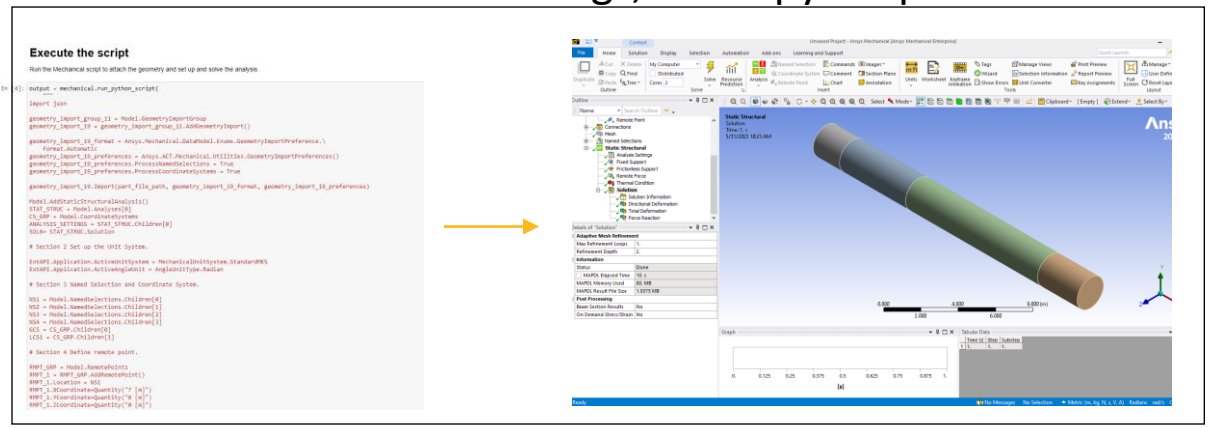


PyMechanical: Access Mechanical through Python

`ansys-mechanical-core`

Remote session

- Based on gRPC:
  - Server = Mechanical
  - Client = PyMechanical
- Send commands as strings, or as .py scripts

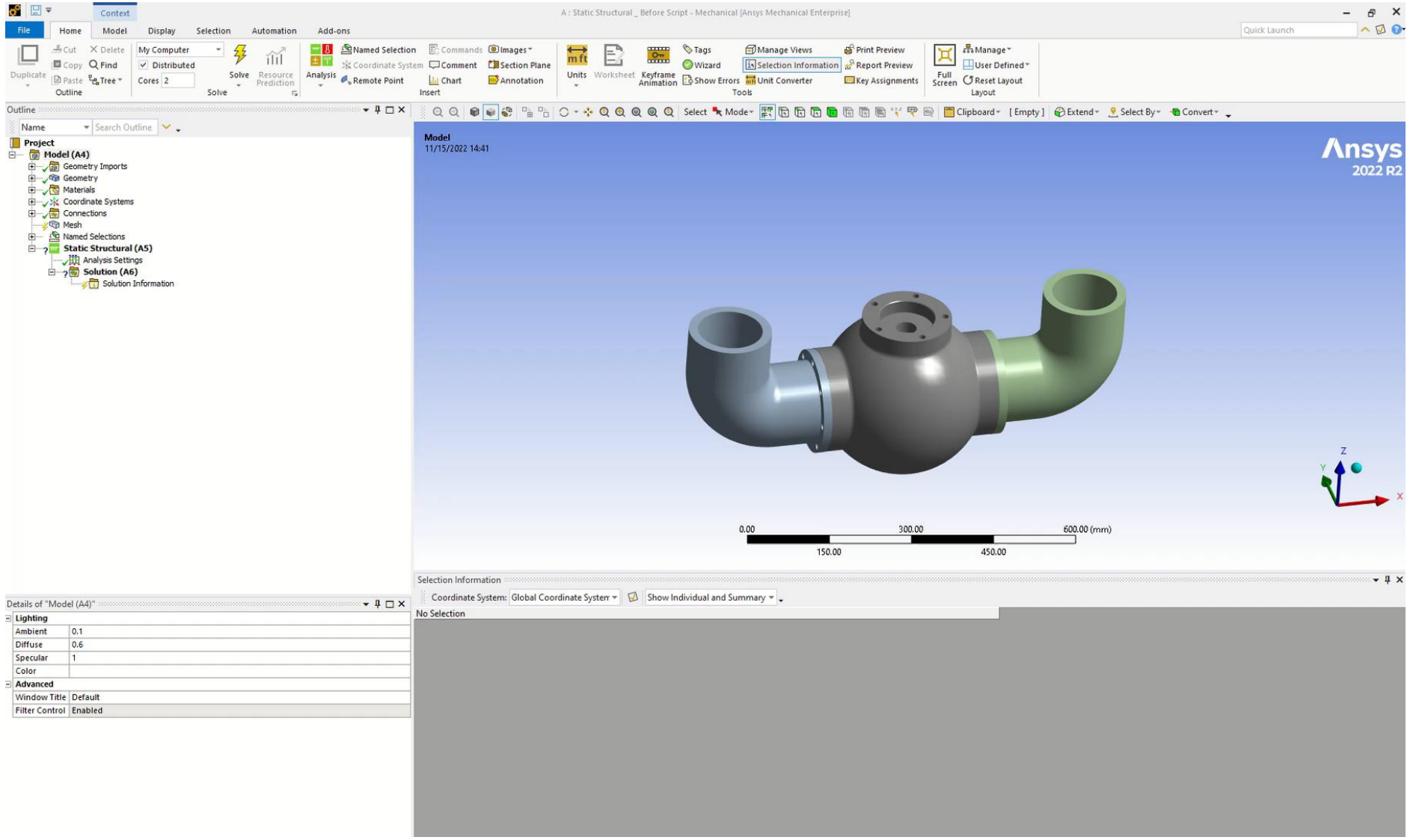


Embedded instance

- Based on Python.NET
- Mechanical object directly loaded into Python memory using Python.NET
- Send commands as if in the Mechanical console

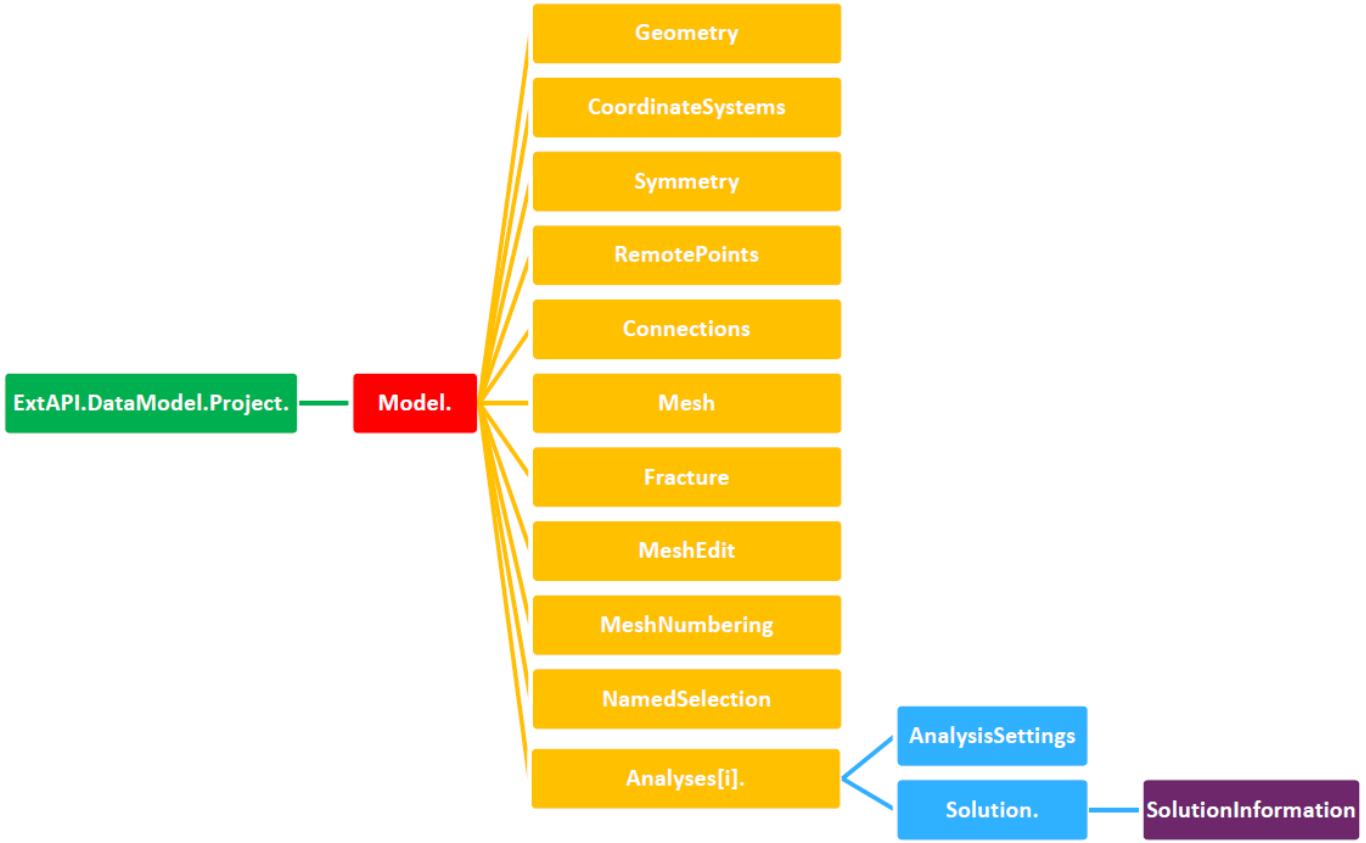
```python
import json

from ansys.mechanical.core import embedding as app
try:
    from ansys.mechanical.core import global_variables
except:
    # No embedding - this import breaks test collection
    global_variables = {}

embedded_app = app.App(version=232)

globals().update(global_variables(embedded_app))

from Ansys.Mechanical.DataModel.Enums import *
from Ansys.ACT.Interfaces.Common import *
# Wrapper functions to shorten overall script
# Function to retrieve geometric body from name
def GetGeoBodyByName(searchName):
    geo = ExtAPI.DataModel.Project.Model.Geometry
    for part in geo.Children:
        for body in part.Children:
            if body.Name == searchName:
                return body.GetGeoBody()
```

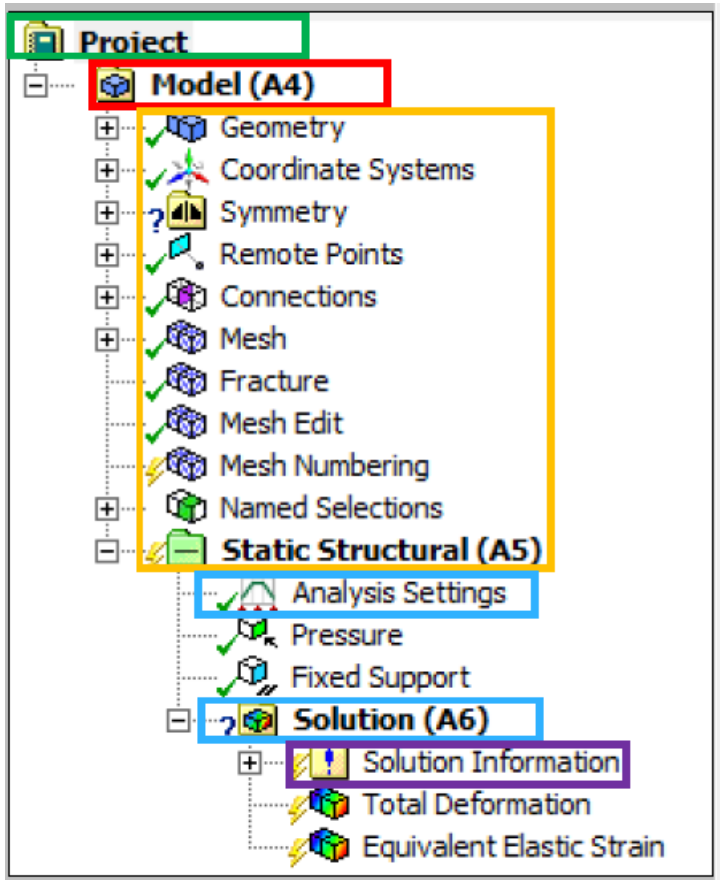
Mechanical scripting demo: complete model setup



Mechanical Automation API

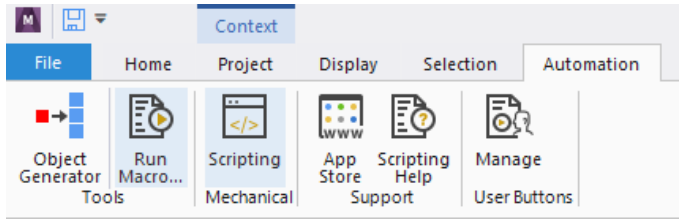
```
# Reference model
model = ExtAPI.DataModel.Project.Model
```

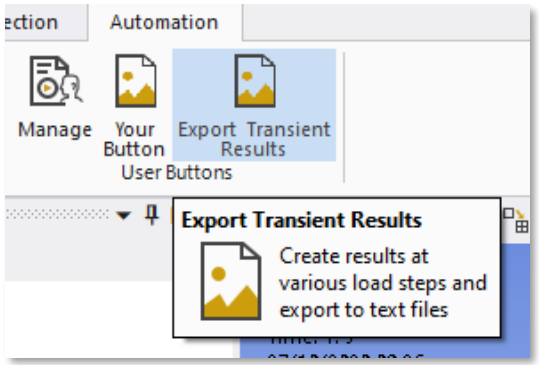




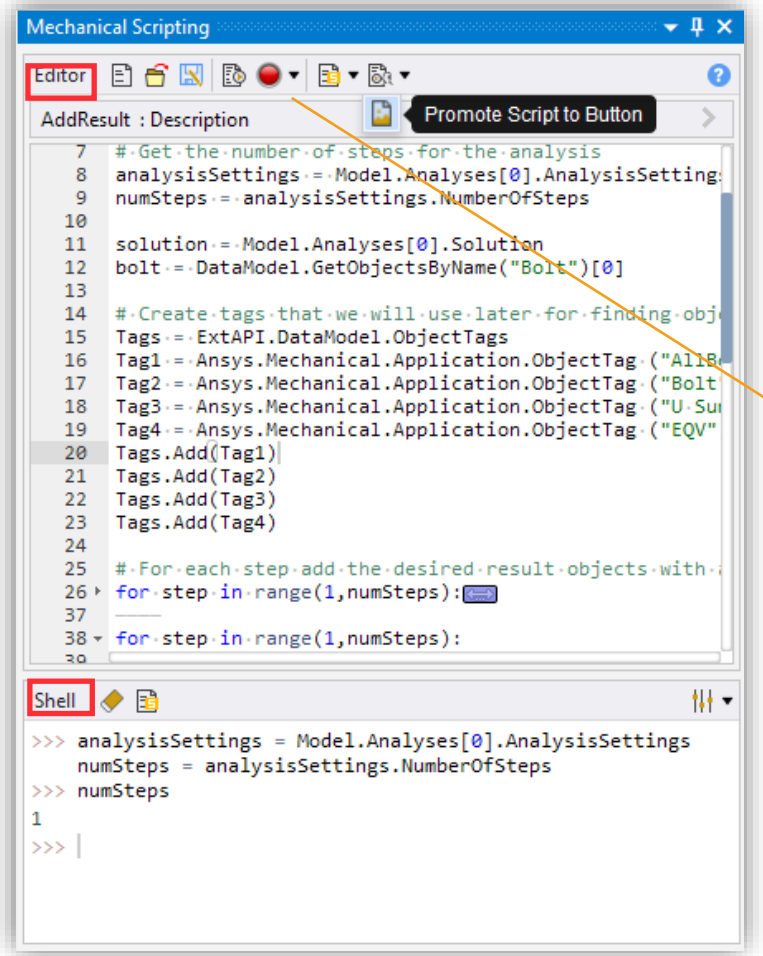
Mechanical Automation Scripting

- Well documented Python APIs with scripting guide and quick start examples
- Shell and multi-line editor to build and debug scripts quickly
- Supports autocompletion and Intellisense
- Ability to easily promote scripts to user buttons
- Recording

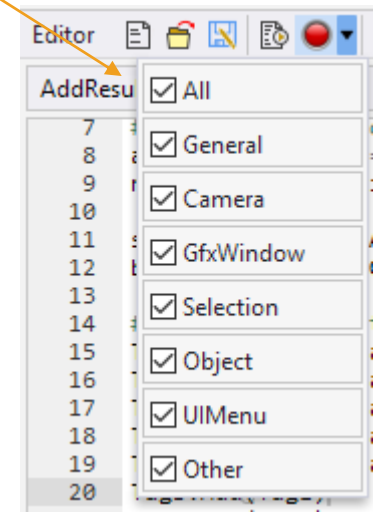




User buttons



Record button with control of how much to journal



PyMechanical demo for embedded and remote versions

This examples illustrates how to setup the same model, once with embedded instance and once with remote version.

Embedded instance

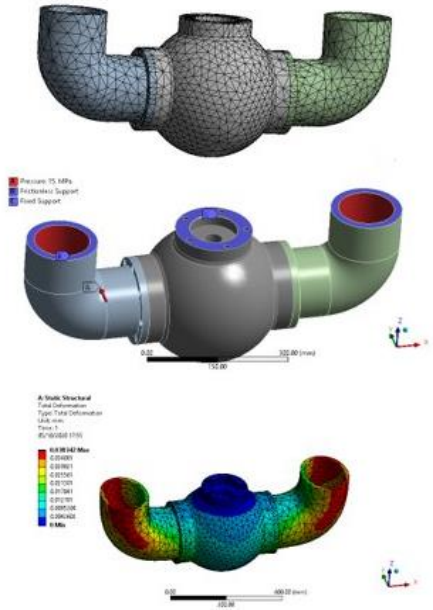
```
import os
from ansys.mechanical.core import App, global_variables

app = App(version=232)
globals().update(global_variables(app)) # update global v

# Add static analysis
analysis = Model.AddStaticStructuralAnalysis()
```

Type commands as if in the Mechanical console

Remote session

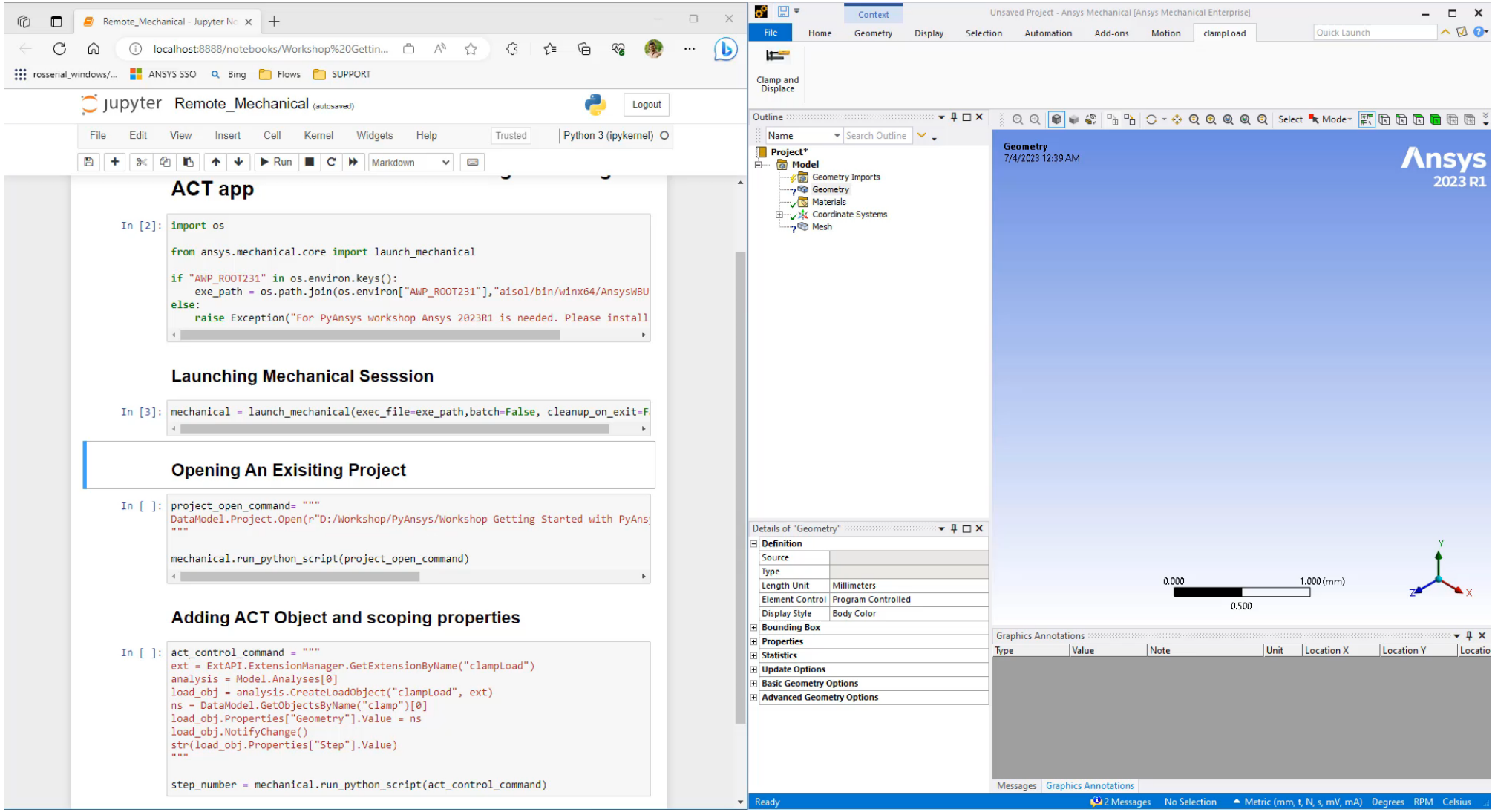
```
import os
from ansys.mechanical.core import launch_mechanical

mechanical = launch_mechanical(batch=True, loglevel="DEBUG")
print(mechanical)
# Check working directory
server_project_directory = mechanical.run_python_script("ExtAPI.DataMode
print(server_project_directory)
# Run mechanical automation script
mechanical_script = os.path.join(work_dir, 'mech_script.py')
result = mechanical.run_python_script_from_file(mechanical_script)
```

Send Mechanical commands as strings or as second Python script



- Import CAD geometry
- Define material assignment
- Define mesh settings
- Define boundary conditions
- Solve
- Post-process
- Export results to text files

Example

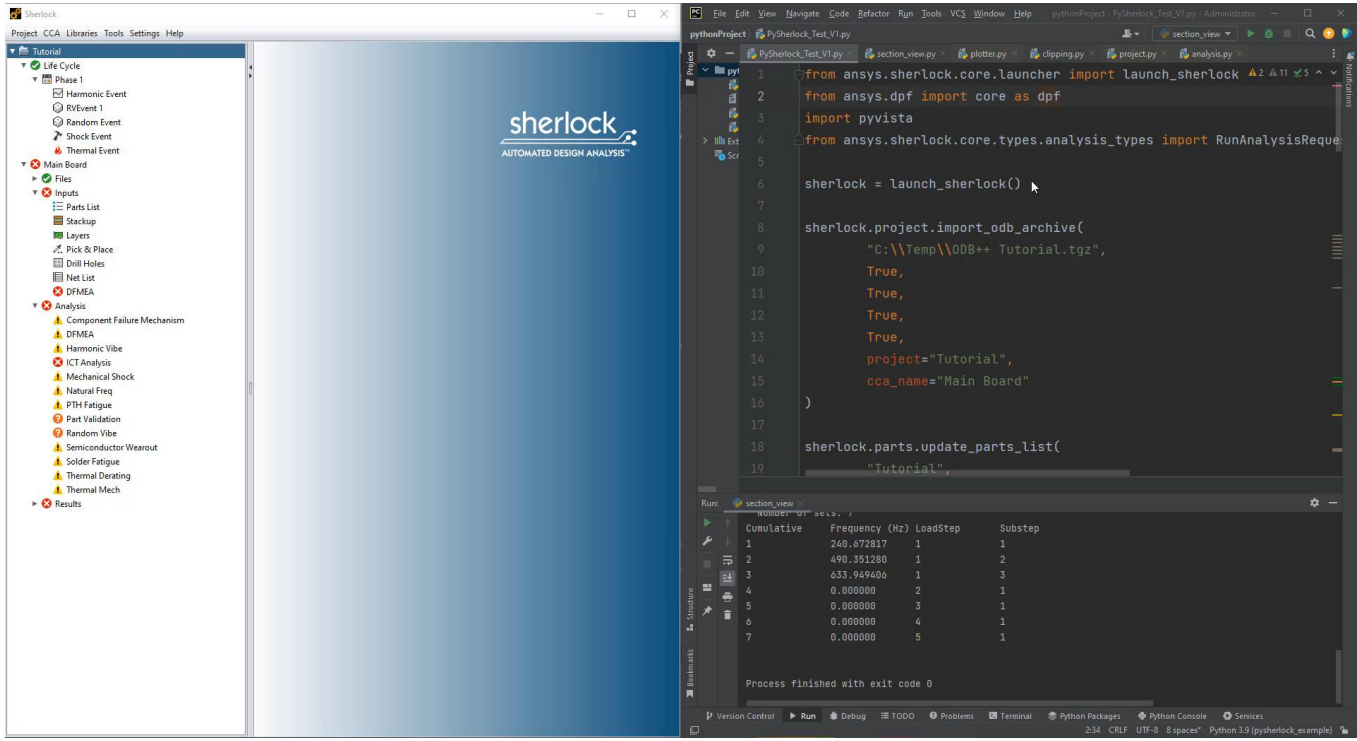


PySherlock

Ansys

PySherlock

- Communicate With Sherlock via CPython
  - Update parts
  - Add vibration events profiles
  - Run analysis
  - Generate report
  - Plot results using pyvista and PyDPF

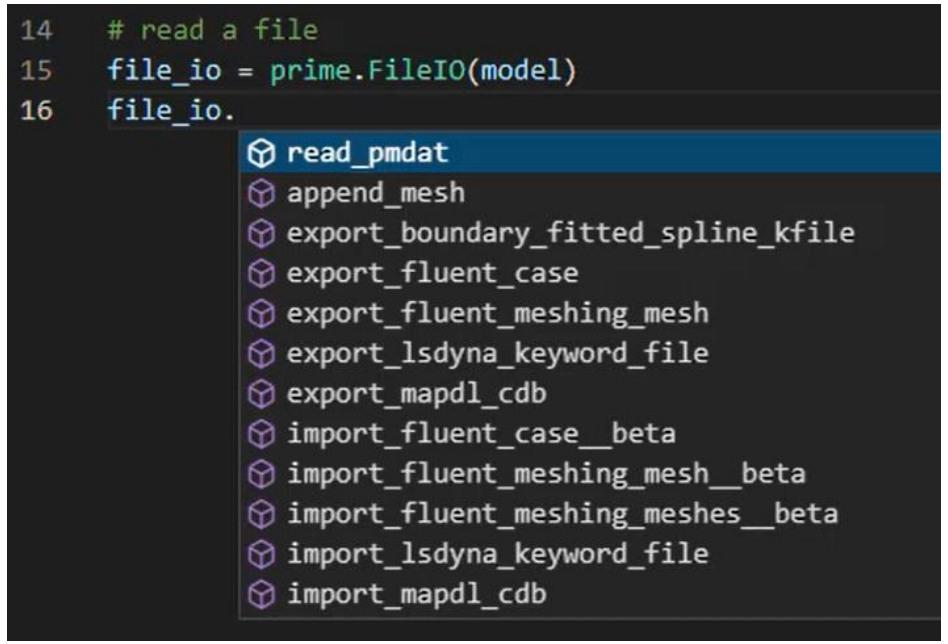


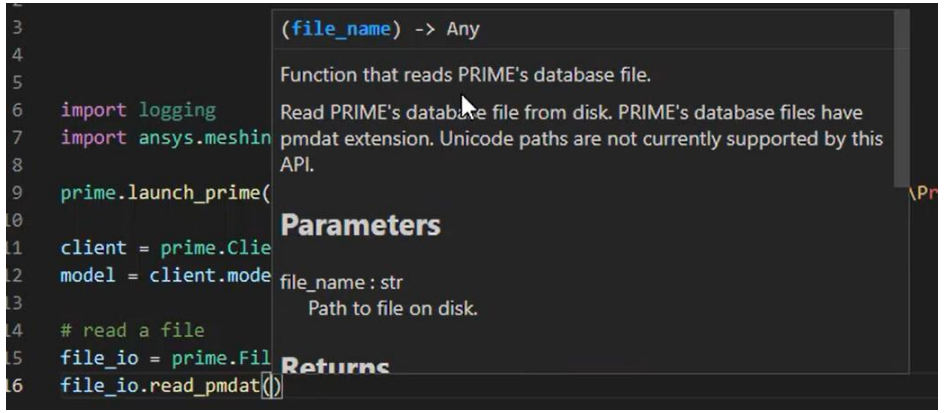
PyPrimeMesh

Ansys

PyPrimeMesh

- PyPrimeMesh lets you use Ansys core meshing technology, embedded across Ansys flagship tools, directly from within your python environment.
- In 2023R1 with PyPrimeMesh, it's possible to:
  - Generate surface and volume mesh of various types using parallel capabilities
  - Apply a range of sizing controls to control mesh distribution
  - Create complex workflows and automation
  - Use wrapping methods to extract and mesh regions
  - Modify connectivity of topology and mesh
  - Import and facet CAD
  - Export to various Ansys solvers

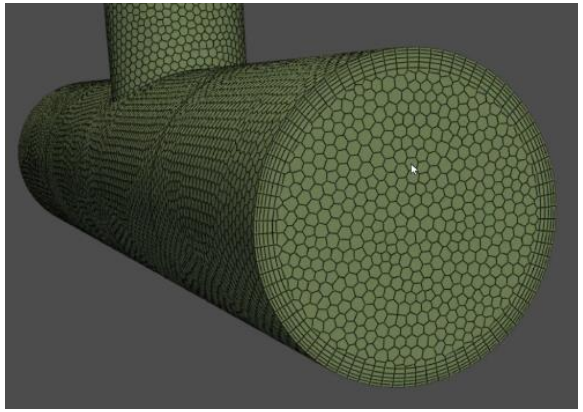


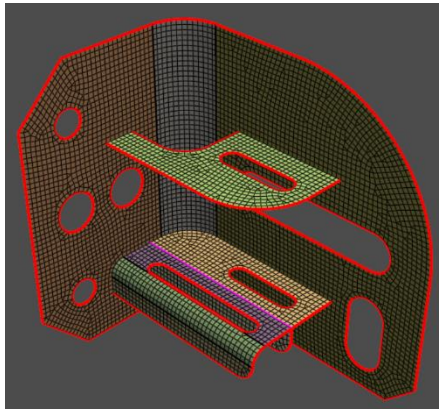


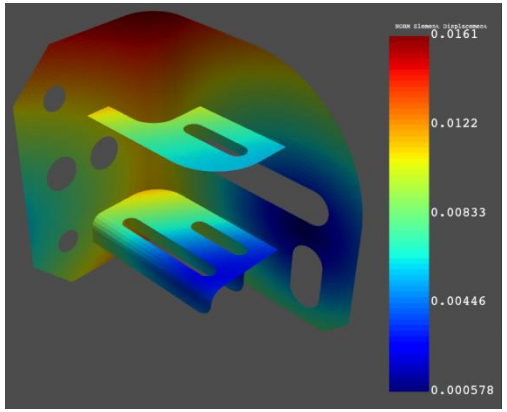
PyPrimeMesh Examples

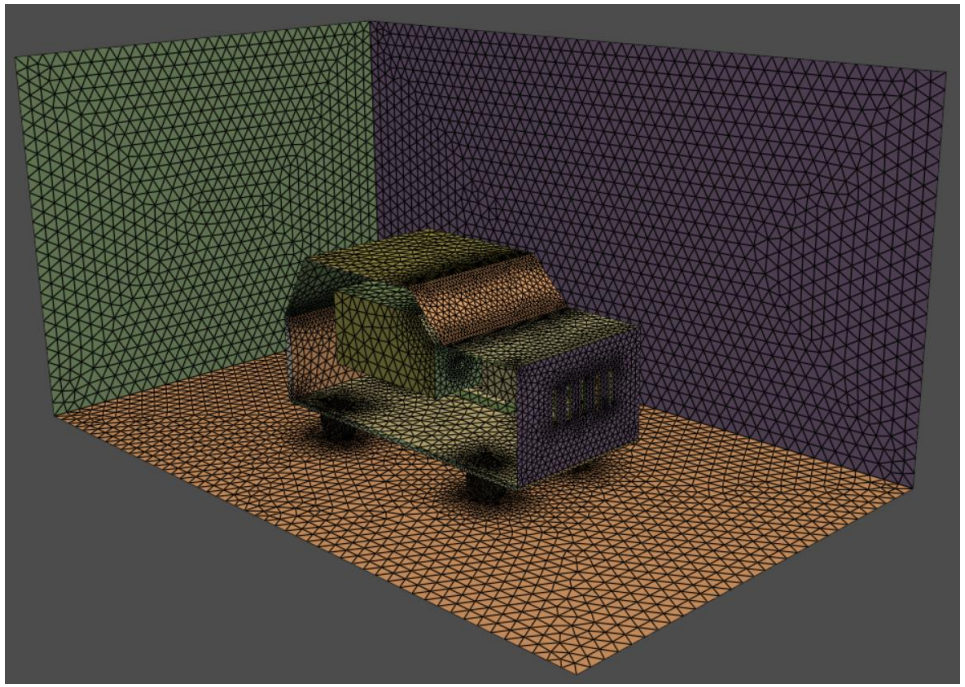
Some examples:

- Wrapping
- Poly mesh with inflation
- Quad shells
- Connecting topology
- …

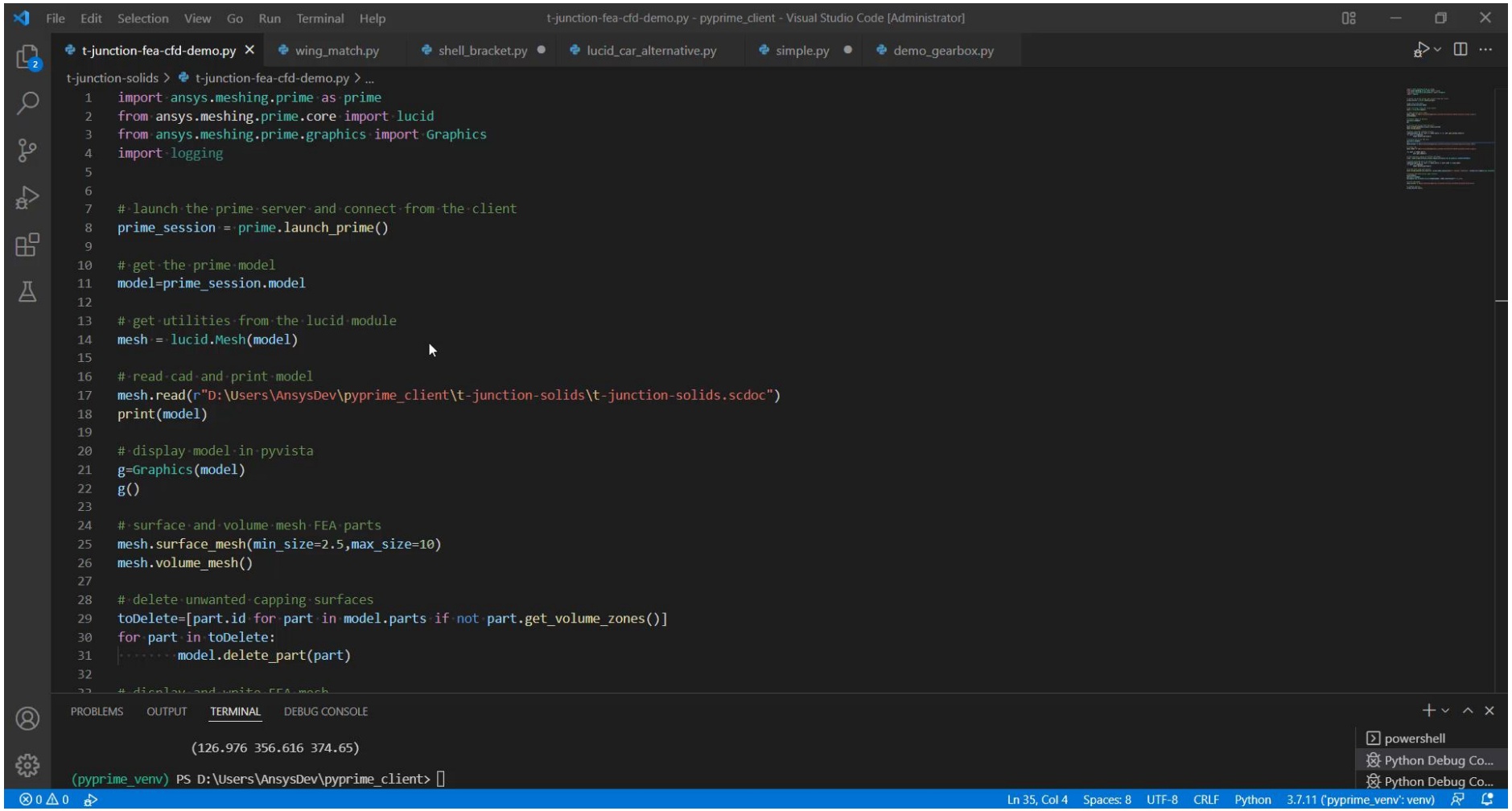








Mechanical / CFD - T-section demo
clean CAD & wrapping mesh processes



Input:
scdoc geometry

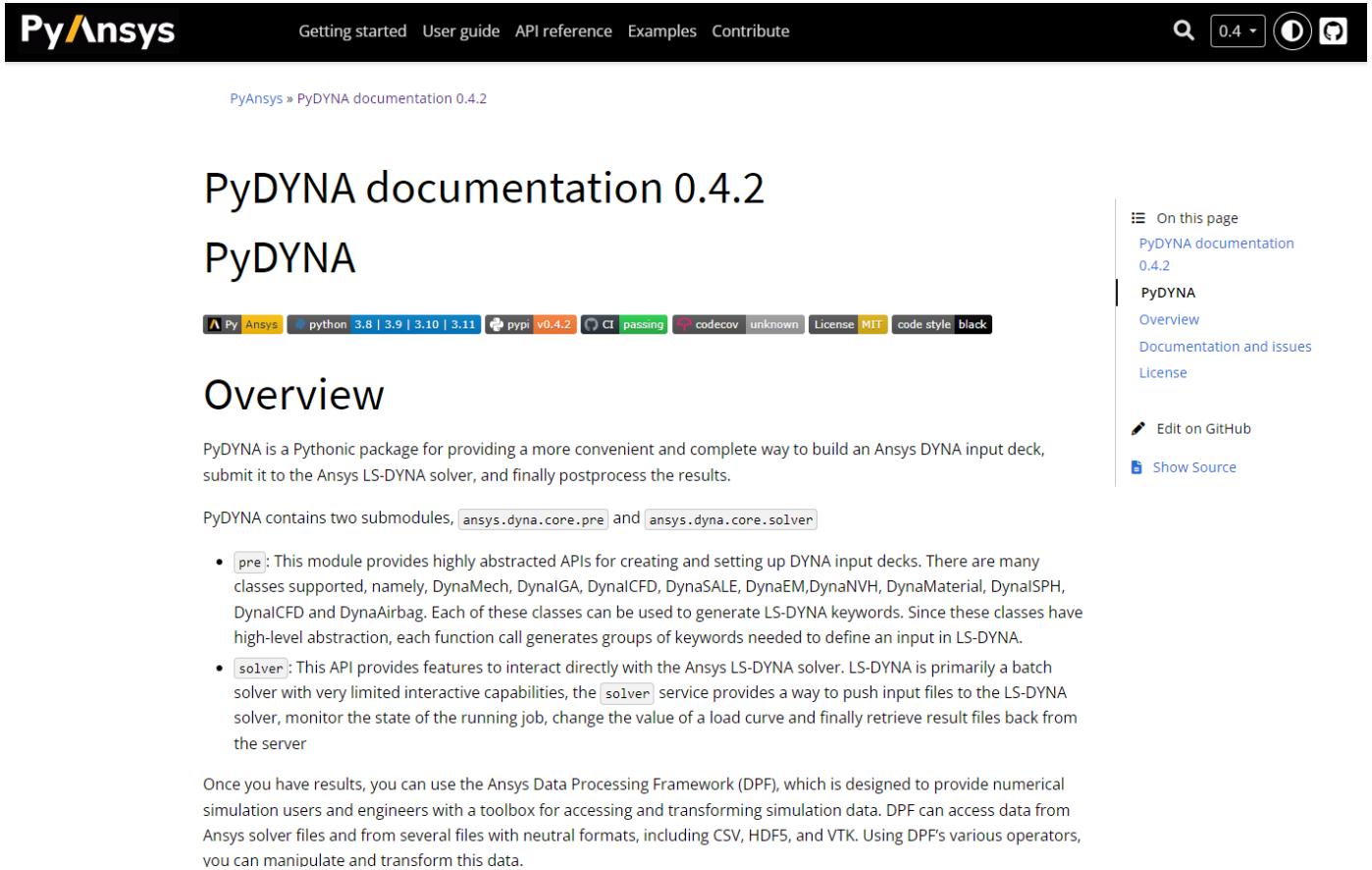
Output:
Mechanical cdb
+
Fluent msh

Reading -> meshing
-> writing using
high level api's

PyDyna

Ansys

PyDYNA is Public !!!



https://dyna.docs.pyansys.com/version/stable/

https://github.com/ansys/pydyna

DYNA input deck generation

Connect to dyna.pre server

```
hostname = 'localhost'
drop_test = DynaSolution(hostname)
fns = []
path = sys.path[0]+os.sep
fns.append(path + "main.k")
fns.append(path + "nodes_elements.k")
drop_test.open_files(fns)
```

pydyna.pre API

CONTROL and DATABASE Cards

```
drop = DynaMech()
drop_test.add(drop)
drop_test.set_termination(0.01)
drop.set_timestep(timestep_size_for_mass_scaled=-4e-9)
drop_test.set_output_database(glstat=1.e-7,matsum=1.e-7,rwforc=1.e-7)
```

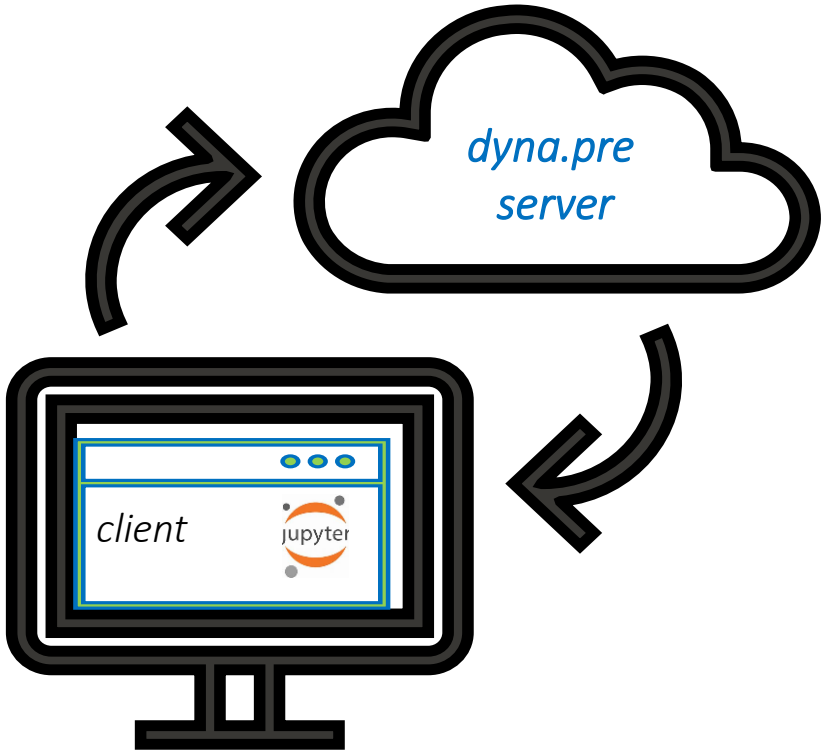
MATERIAL Cards

```
mat_1 = MatElastic(mass_density=1.3e-9,young_modulus=4100.,poisson_ratio=0.29
mat_2 = MatElastic(mass_density=1.57e-9,young_modulus=25000.,poisson_ratio=0.
mat_3 = MatElastic(mass_density=1.0e-9, young_modulus=3000., poisson_ratio=0.
mat_4 = MatPiecewiseLinearPlasticity(mass_density=8.30e-09,young_modulus=1100
                                     poisson_ratio=0.34,yield_stress=280.
                                     tangent_modulus=1150.)
```

Assign Material and Section to Parts

```
Bottom_Top = SolidPart(1)
Bottom_Top.set_material(mat_1)
Bottom_Top.set_element_formulation(SolidFormulation.CONSTANT_STRESS_SOLID_ELE
drop.parts.add(Bottom_Top)
```

DYNA input deck generation



Easily deployable containers or standalone servers

RIGIDWALL , INITIAL VELOCITY and GRAVITY

```
rigidwall = RigidwallPlanar(Point(-1.8581,-32.039,2.904), Point(-2.0,-32.039,
                            coulomb_friction_coefficient=0.1)
drop.add(rigidwall)

saset = range(1, 25)
velocity = drop.initialconditions.create_velocity(PartSet(saset),velocity=Vel

g = Gravity(dir=GravityOption.DIR_X, load=Curve(x=[0.005, 1e18], y=[1, 1]))
drop.add(g)
```

TIED CONTACT between Part Sets

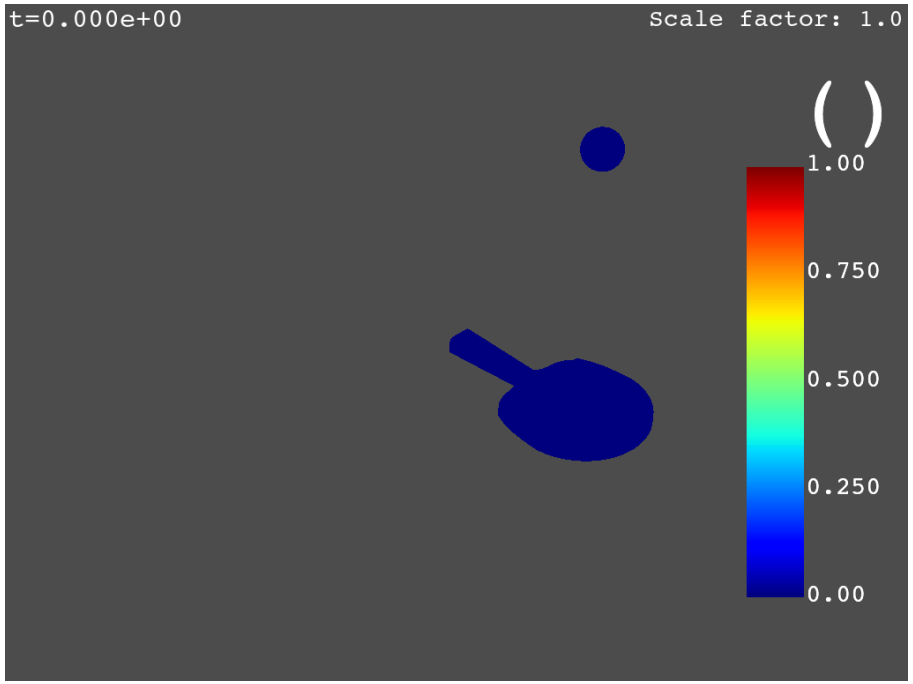
```
contact = Contact(category=ContactCategory.SHELL_EDGE_TO_SURFACE_CONTACT
                  ,type=ContactType.TIED)
surf1 = ContactSurface(PartSet([5]))
surf2 = ContactSurface(PartSet([9,4]))
contact.set_slave_surface(surf1)
contact.set_master_surface(surf2)
drop.add(contact)
```

Save the file

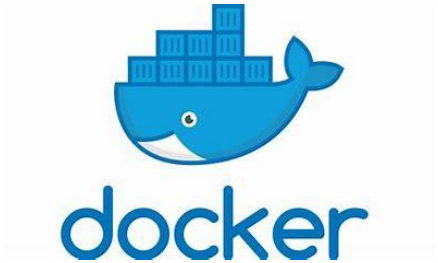
```
drop_test.save_file()
```

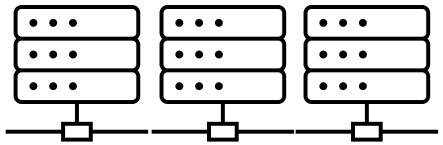
Interactive Solver

```
# my docker files are under
#D:\PYDYNA_BETA_V.0.1\pyDyna\src\ansys\dyna\core\solver
import ansys.dyna.core.solver as  solver

dyna = solver.DynaSolver('localhost','5000')
dyna.push(r'D:\pyDyna_Examples\DROP_BINOUT\tt.k')
dyna.start(4)
dyna.run('i=racket_inp.k ncpu=4 jobid=tt')
```




```
while t_step < Tend:
    dyna.pause()
    (c,t) = dyna.time()
    c1000_1,v1000_1 = dyna.node(100000)
    c2000_1,v2000_1 = dyna.node(200000)
    c3000_1,v3000_1 = dyna.node(300000)
    cx1000_1 = c1000_1[0]
    cz1000_1 = c1000_1[2]
    cx2000_1 = c2000_1[0]
    cz2000_1 = c2000_1[2]
    cx61_1 = c3000_1[0]
    cz61_1 = c3000_1[2]
    vx1000_1 = v1000_1[0]
    vz1000_1 = v1000_1[2]
    xpos10_1=vx1000_1

    ang_ball=(math.atan(vx1000_1/vz1000_1))
    vx_plate=cx2000_1-cx61_1
    vz_plate=cz2000_1-cz61_1

    ang_plate=(math.atan(vx_plate/vz_plate))
    diff_ang=ang_ball-ang_plate
    distz=cz1000_1-cz61_1
    t_imp=abs(distz/vz1000_1)
    rot_vel=diff_ang/t_imp

    if cz1000_1 < 60:
        dyna.setlc(30,rot_vel)
    else:
        dyna.setlc(30,no_vel)

    dyna.setlc(20,xpos10_1)
    t_step = t_step + step_size
    t_sol += [t_step]
    n_step+=1
    try:
        dyna.resume(time=t_step)
    except:
        print('LSDYNA run terminated')
```



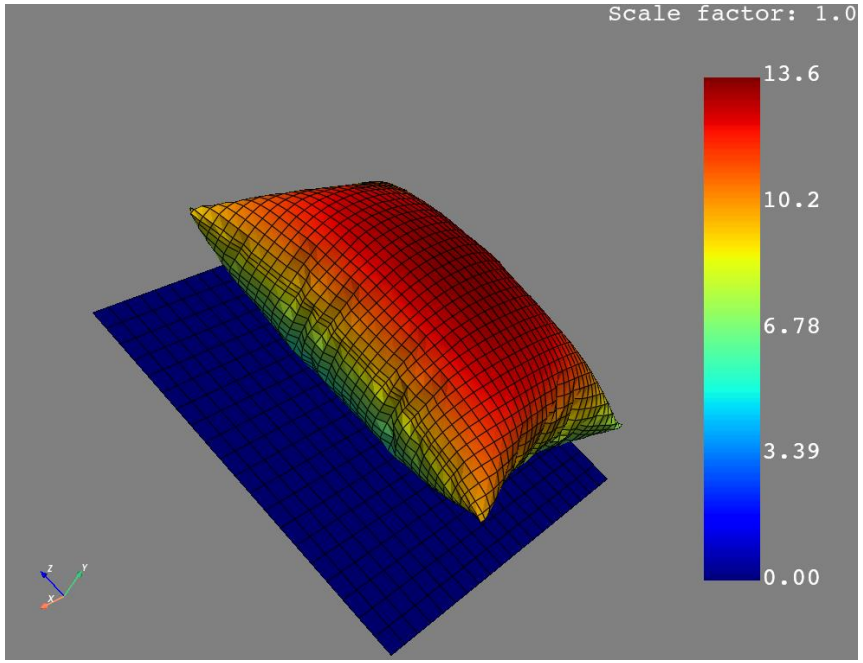


MPP *DYNA*

Read the data from the server and post process

Plot the displacement of select parts

```
model = dpf.Model(data_sources=ds,server=server)
print(model)

part_id = [6,9]

stre = dpf.operators.result.stress(time_scoping=[11])
stre.inputs.data_sources.connect(ds)
stre.inputs.requested_location.connect("Nodal")
stre.inputs.time_scoping.connect(range(12))
stre.connect(25, part_id)
fieldsStr = stre.outputs.fields_container()
shell_layer_extract = dpf.operators.utility.change_shell_layers()
shell_layer_extract.inputs.fields_container.connect(fieldsStr)
shell_layer_extract.inputs.e_shell_layer.connect(0)
fields_top = shell_layer_extract.outputs.fields_container_as_fields_container()
field = fields_top.get_field({'time':11})
field.plot(notebook=False)
```



End to End - Battery Characterization App

Create Cell Geometry
pyGEOMETRY

Mesh Cell Geometry
pyPRIME

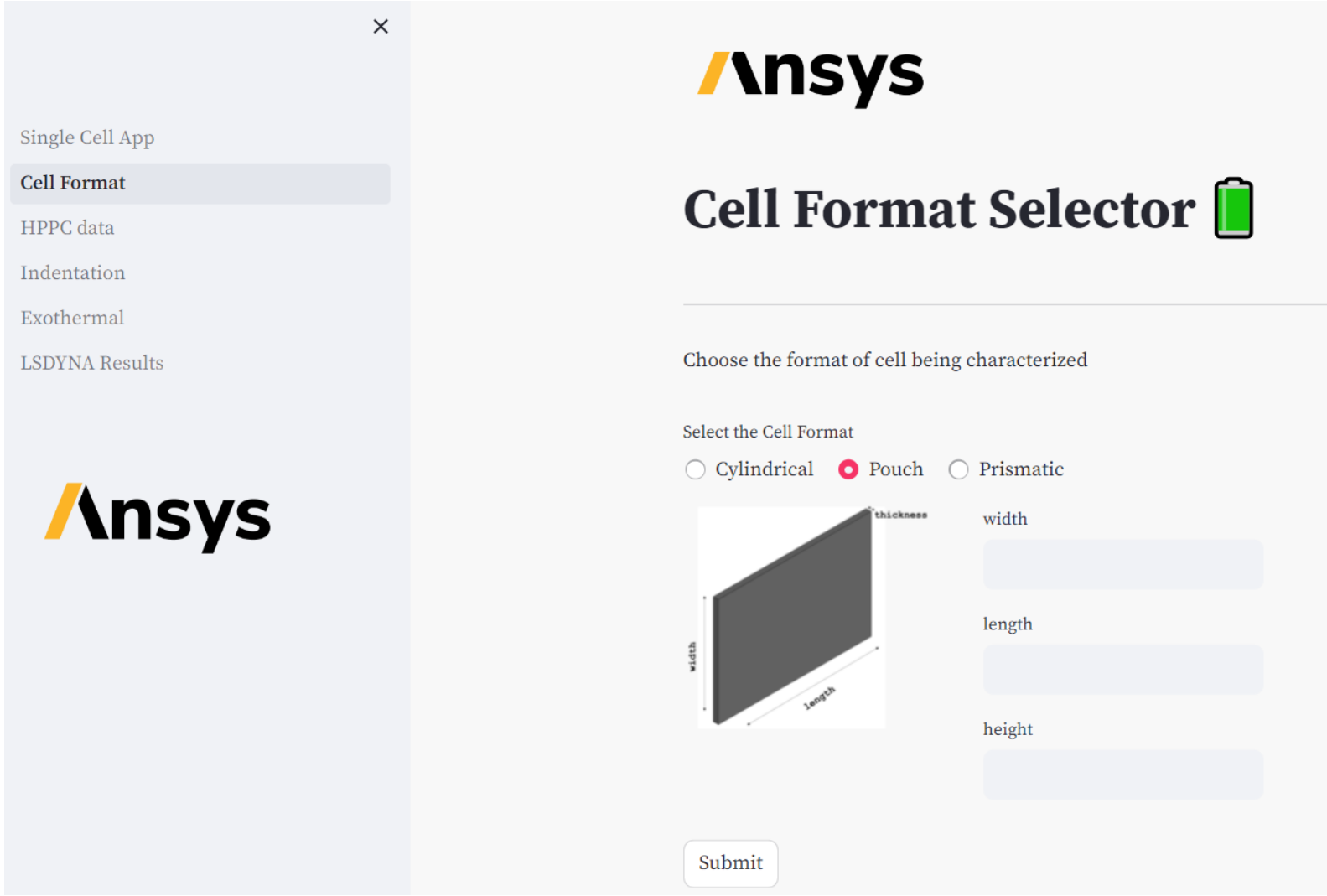
Extract Randles Circuit Parameters from HPPC Test Data
Standalone Python Script

Extract Stress Strain curve from Indentation Test
Standalone Python Script

Setup Dyna Input Deck
pyDYNA.pre

Solve
pyDYNA.solver

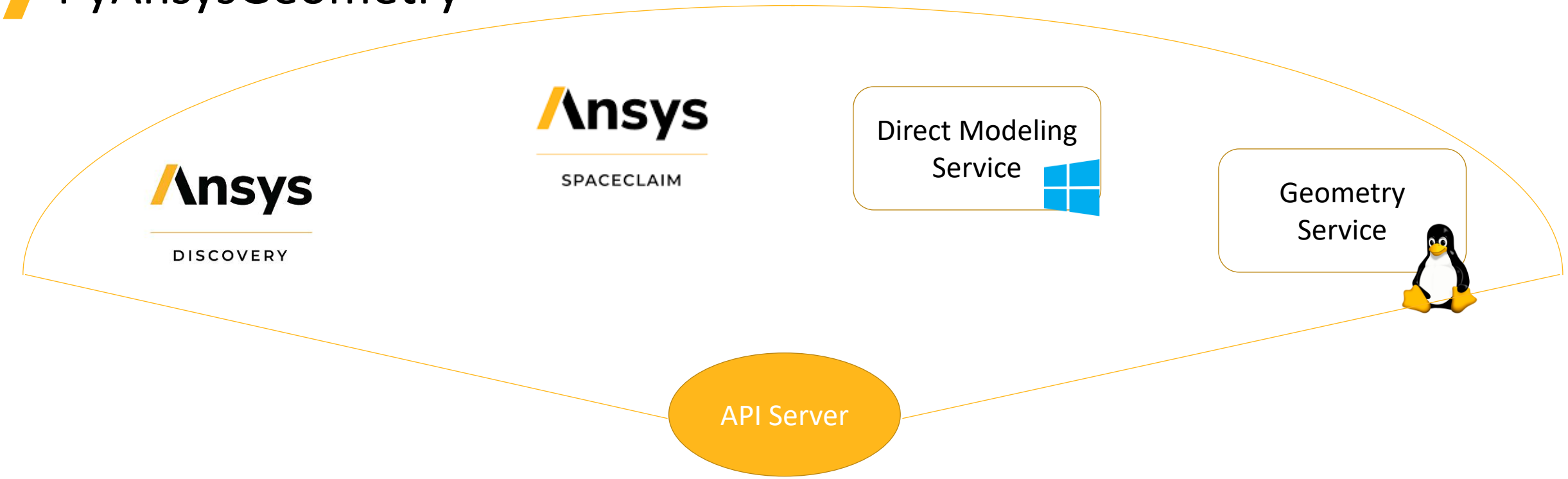
Post Process
pyDPF



PyAnsysGeometry

Ansys

PyAnsysGeometry



PyAnsys

Client-side visualization

- Tessellation allows to visualize geometries on client-side

```
[5]: # Create a new design
design = modeler.create_design("AdvancedFeatures_SpurGear")

# Extrude your sketch
dummy_gear = design.extrude_sketch("SpurGear", sketch, Distance(200, UNITS.mm))

# Let's see how it looks like
design.plot()
```