

# Waltham Users Forum: Unleashing the Power of PyAnsys

Sandeep Medikonda

September 20<sup>th</sup>, 2023

# / Agenda

- What is PyAnsys?
- Why use it & innovation offered?
- What's Available?
- Popular PyAnsys Workflows
  - Machine Learning Example
  - WebApps
- How to install it?
- Documentation, Help & Other Resources
- Conclusions



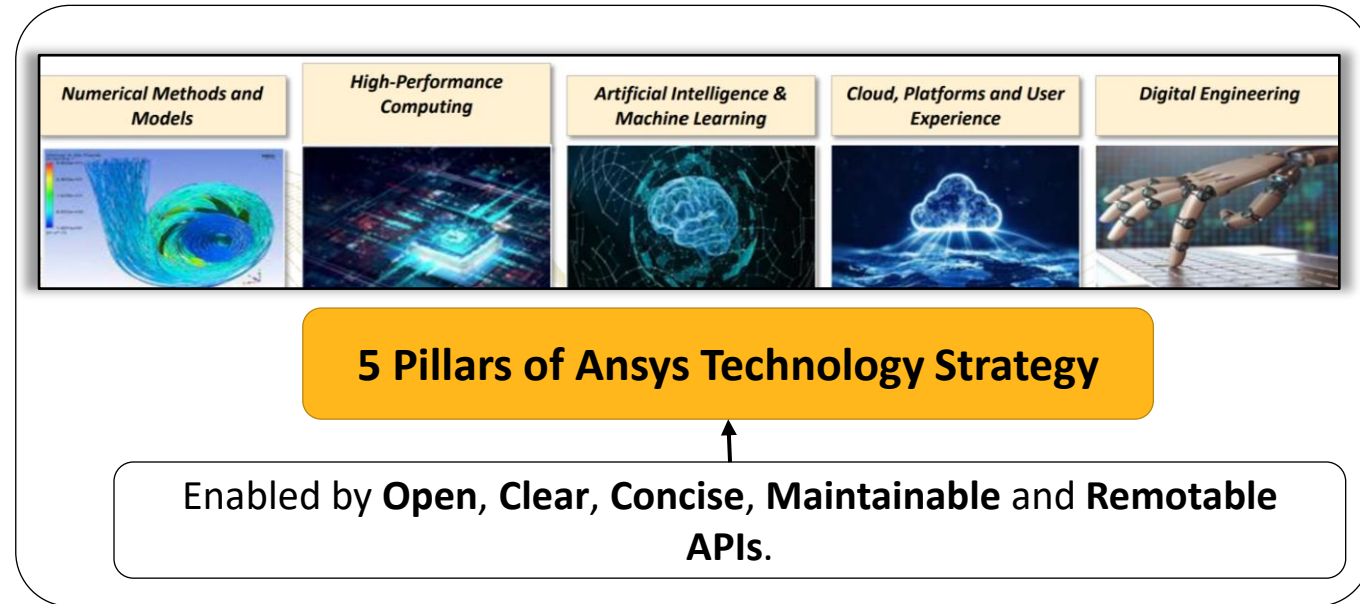
# / Need for APIs?

## Evolving Ansys Customer Needs:

- Democratization of simulation (including non-engineers)
- Accelerate predictable insights
- Connect Ansys technologies to wider enabling technologies such as AI/ML

## Ansys Assessment:

- Requirements can be classified into 5 Pillars: **Numerical Solvers, High-Performance Computations, AI/ML, Cloud** and **Digital Engineering**.
- Need for strong and open APIs



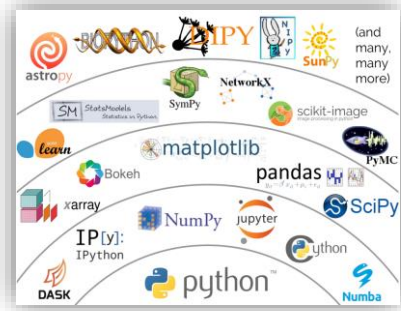
It's all about increasing the value of our tools for customers...

# What is PyAnsys?

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

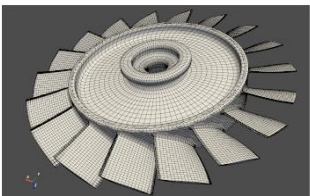


Python + Ansys = PyAnsys



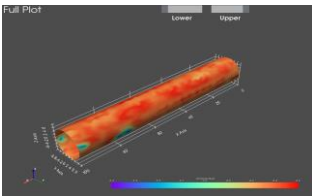
PyMAPDL

Access MAPDL simulation and solver



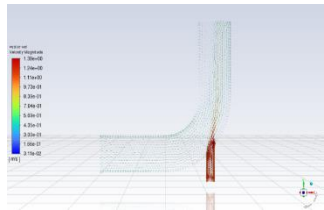
PyAEDT

Access Scripting functionalities of the AEDT Suite



PyFluent

Pythonic interface to Ansys Fluent



More...

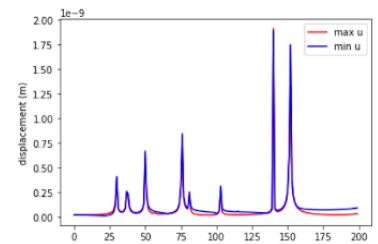
Visit... [docs.pyansys.com](https://docs.pyansys.com)

Post-processing libraries

DPF-Core

Read and manipulate finite element data with powerful, scalable operators

DPF-Post



Extract simulation data with a streamlined post-processing API

# Putting the Py in PyAnsys

- Py is short for python, the language, not the snake
  - nor pie which is a shame.
- Python is becoming the *lingua Franca* of engineering software languages especially data investigation (e.g AI/ML).

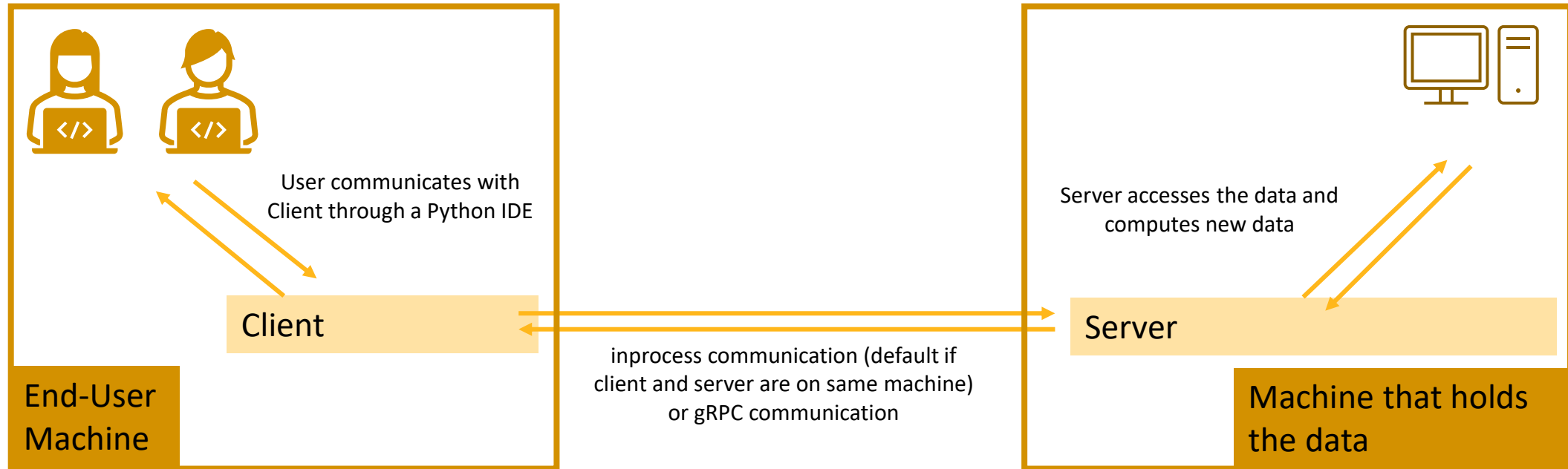


## Most Popular Programming Languages 2000



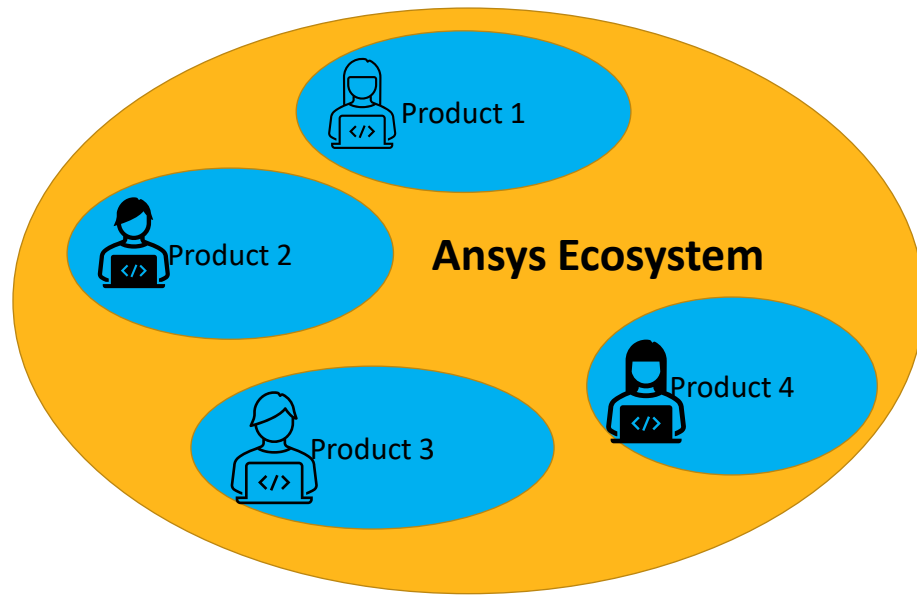
# Client/Server architecture

- [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 machines. Below is a typical configuration:

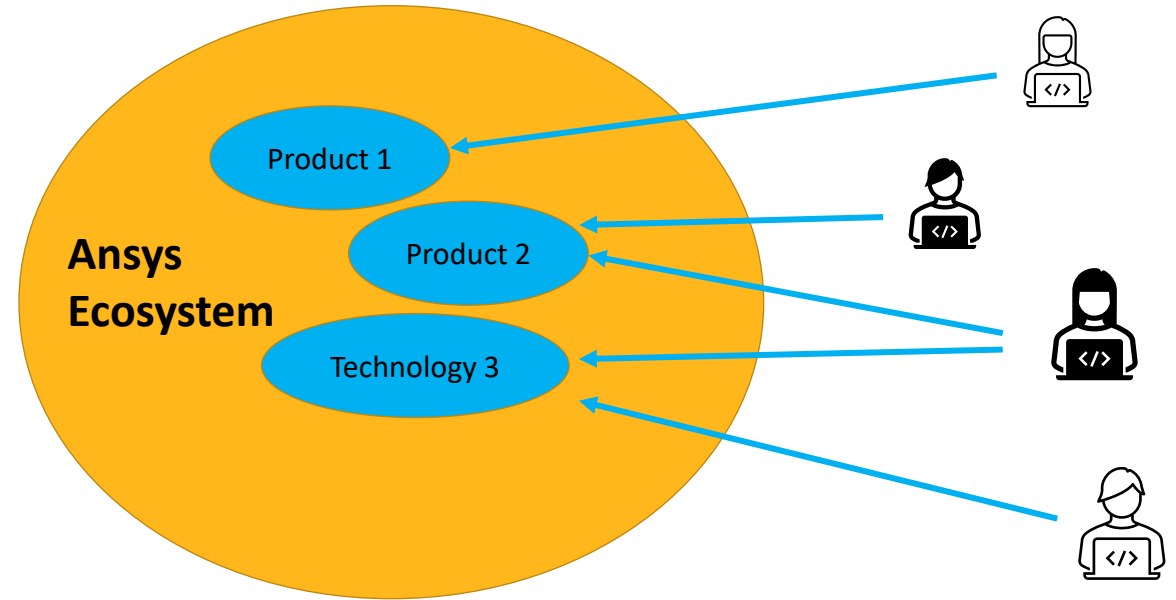


- Where and how to find the files?
  - The Client part comes is provided through Python packages (pip install)
  - The Server part is shipped with the standard Ansys installation

# Two possible approaches to scripting



Product scripting

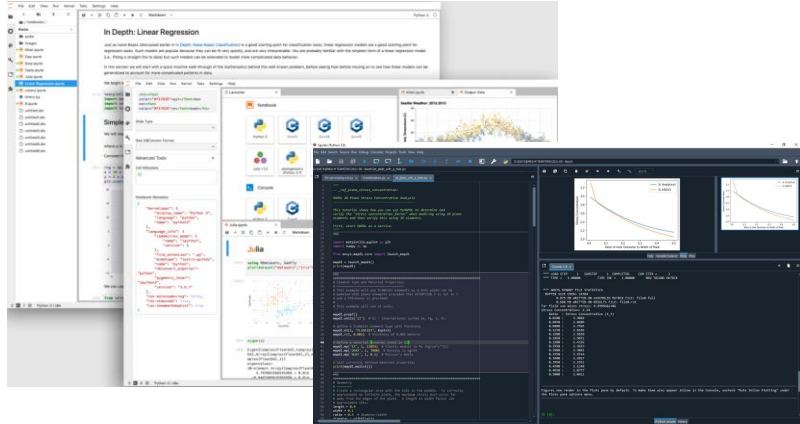


PyAnsys scripting



# Scripting from within the customer tools – PyAnsys

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



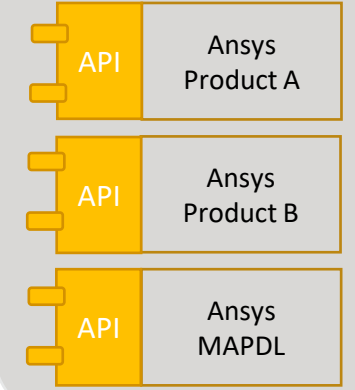
Possibility to use other useful Python tools



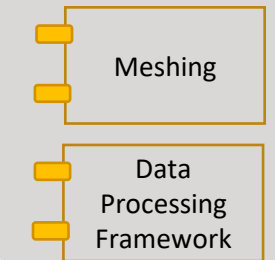
PyAnsys technology



Products



Technology components



Ansys Technology

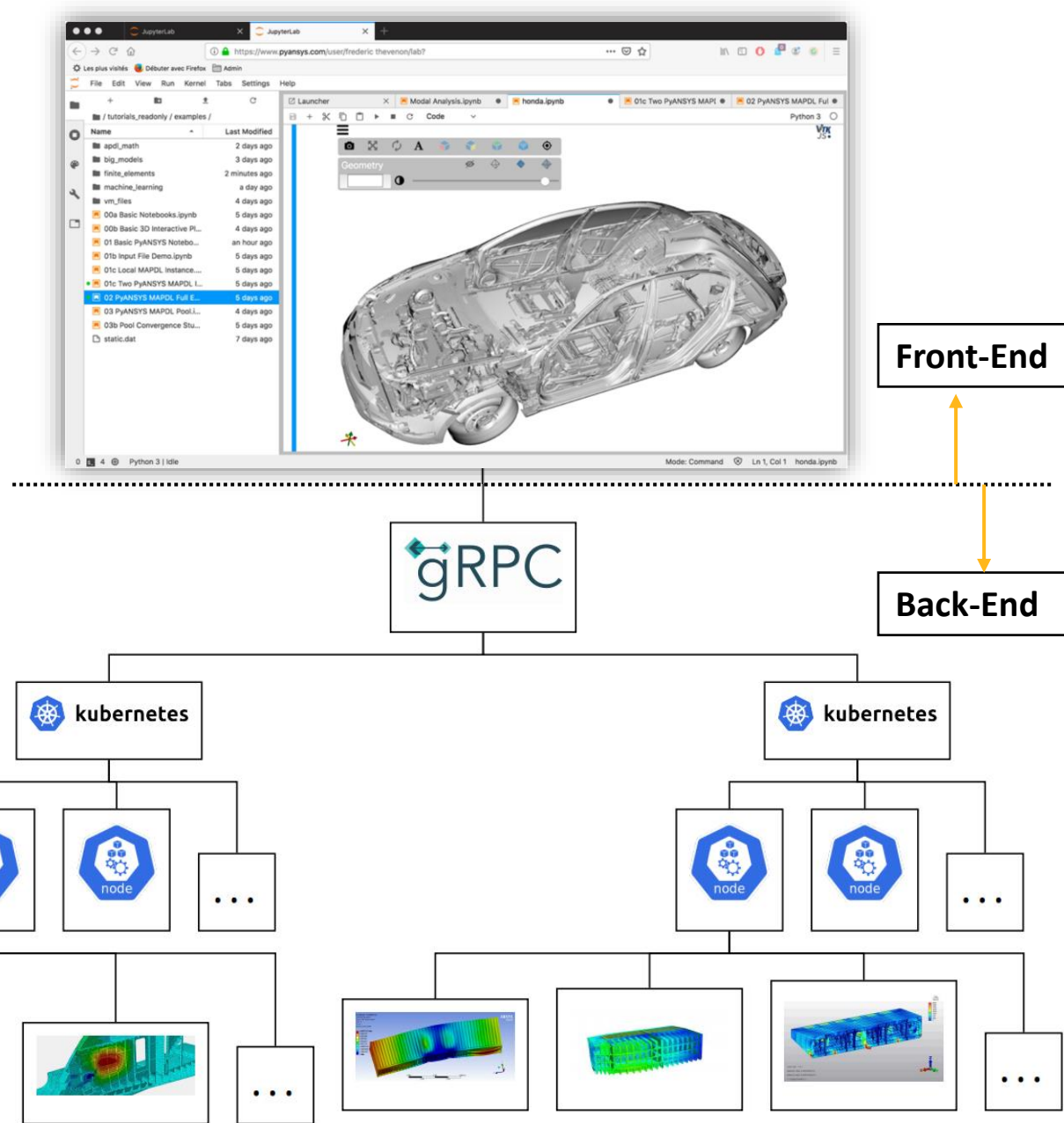
Customer tools



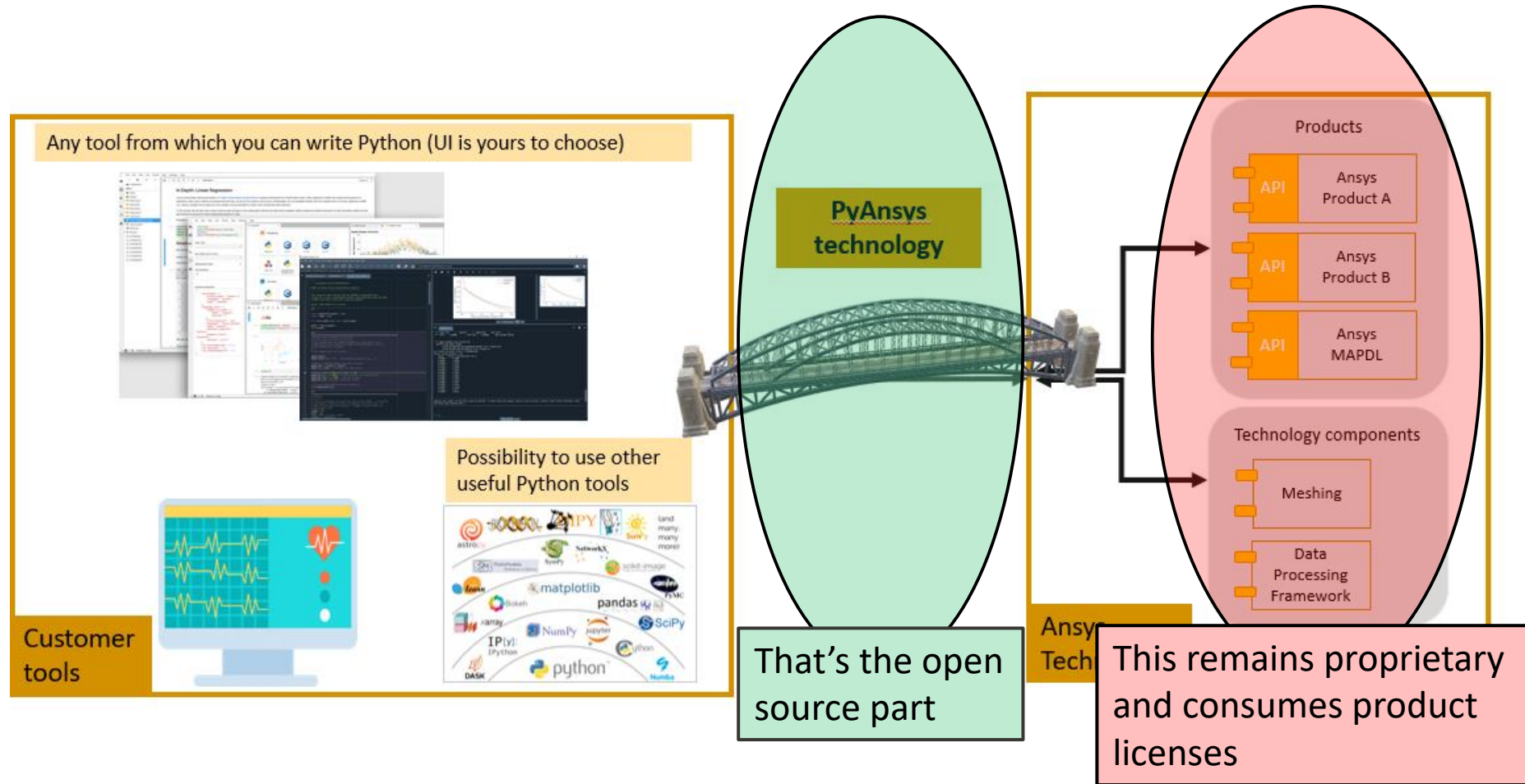


# Why use it & Innovation offered?

- 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
- Built on cutting edge technologies such as:
  - gRPC
  - Kubernetes Framework (or any HPC)
  - Headless Docker Containers



# 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.

# PyAnsys: What's available?

## 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

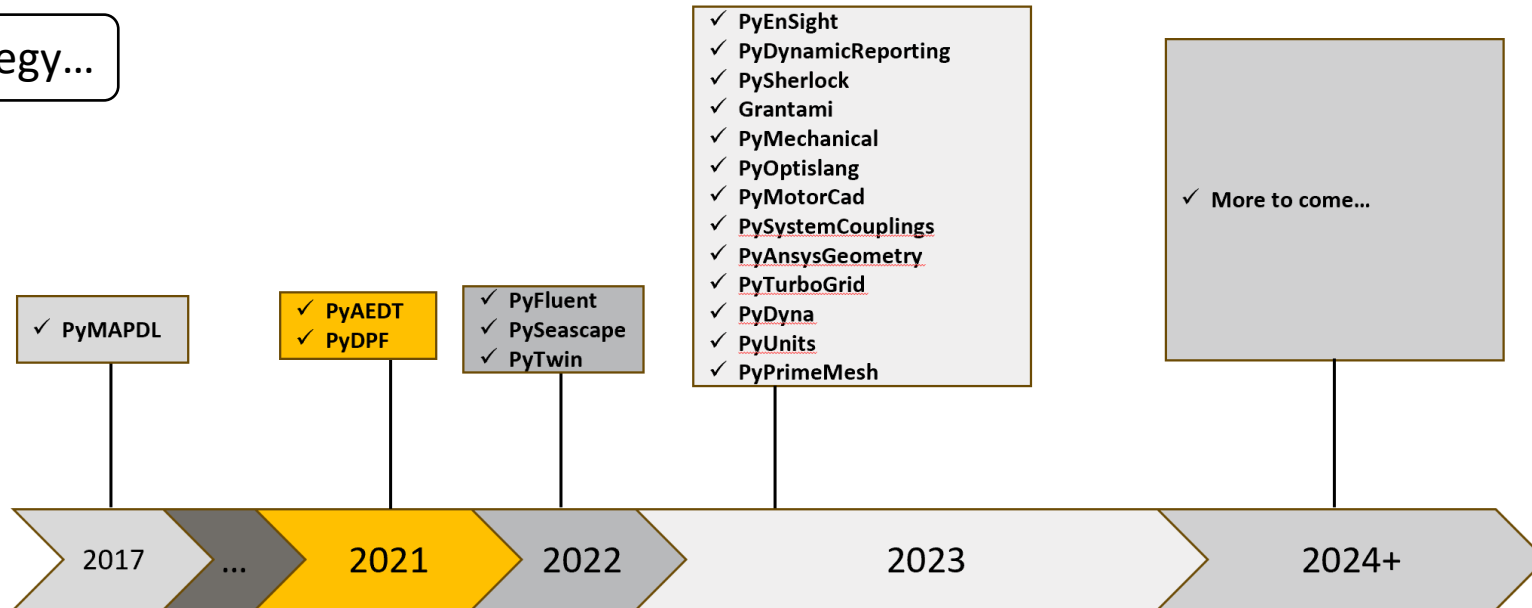
## Post-processing libraries

**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

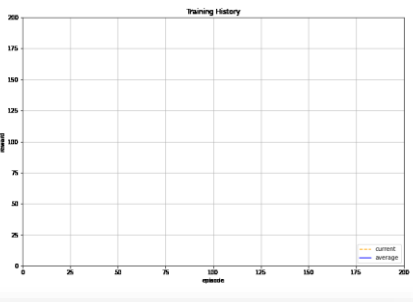
**PyEnSight**  
Pythonic interface to EnSight, the Ansys simulation postprocessor

## 5-year strategy...



Keep an eye on...  
[docs.pyansys.com](https://docs.pyansys.com)

# Key workflows where Ansys customers are using PyAnsys



### PyFluent integrated with Machine Learning Ecosystem of Python

- Boundary Condition setup, Initialization and Iterations are performed using PyFluent API
- Response is extracted using PyFluent API and stored in a 2D Numpy Array
- Supervised Learning using different algorithms: Linear Regression, XGBoost, TensorFlow (deep neural network)

DOC using Numpy

Cold Inlet Velocity	Hot Inlet Velocity
0.1	0.8
0.2	1.0
0.3	1.2
0.4	1.4
0.5	1.6
0.6	1.8
0.7	2.0

3D Interactive Response Surface using Plotly

Parity Plot using matplotlib

Setup, Solution and Post-processing using PyFluent

Input Parameters:

- Cold Inlet Velocity
- Hot Inlet Velocity

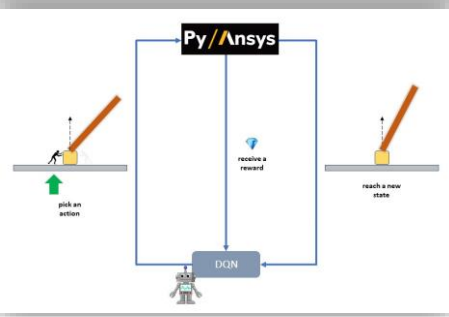
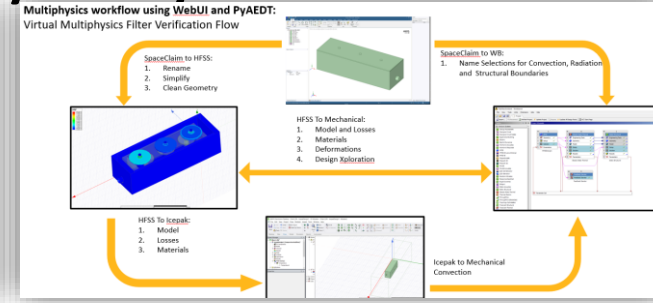
Output Parameters:

- Average Temperature at Outlet

All tasks of this workflow are executed within a single Jupyter Notebook

### PyPrime - PyFluent - PyMAPDL

Geometry      Flow Pattern



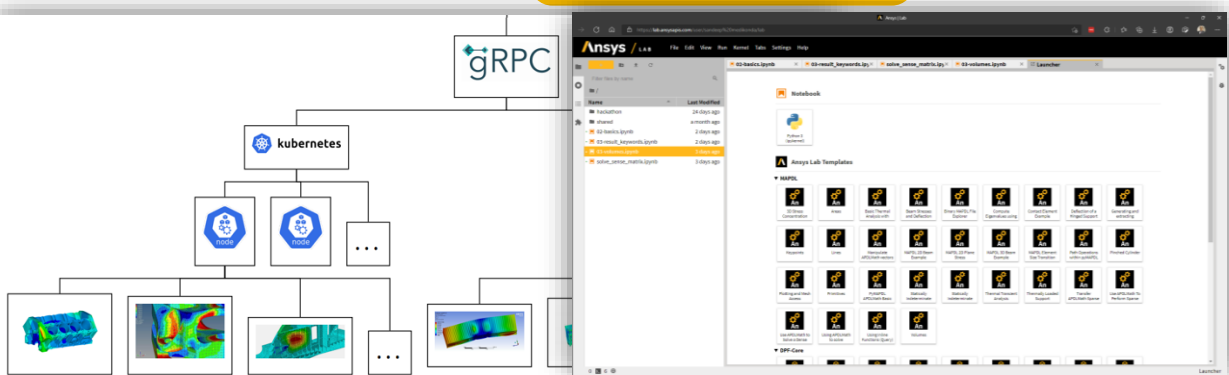
Machine Learning Problems

Multi-Physics workflow Automation



HPC Deployment

Web Applications



#### PyAnsys DPF in a Dash App

This app allows plotting results from built-in variables

Select Example:

Select Time / Frequency:

Select Component:

Plot Results

#### PyAnsys MAPDL in a Dash App

Design your solar array! (not really, dont use these assumed properties)

Number of Panels:

Panel Width (m):

Panel Height (m):

Panel Core Thickness (m):

Hide / Show Additional Options:





# Angle Bracket Optimization



*Angle bracket has “dimple”  
to increase stiffness*

houseofthepworths.com

# ML Example: Topographical Optimization using Reinforced Learning



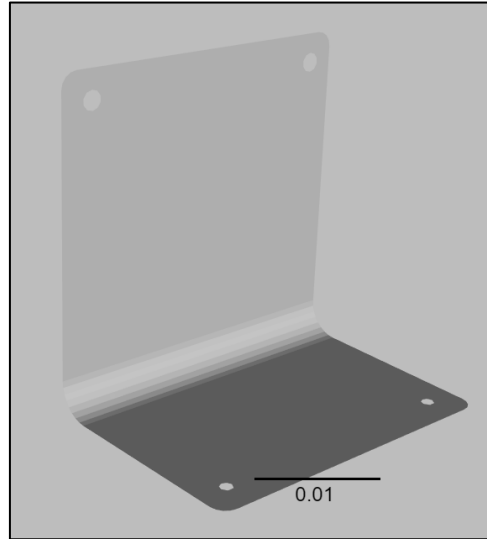
## MDP Description

- Action:
  - morph point
  - morph direction
- State:
  - morph extent
- Reward:
  - Increase freq/stiff
  - penalty for distortion



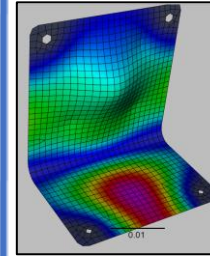
## RL Algorithm

- Find sequences of morphing operations that maximize stiffness

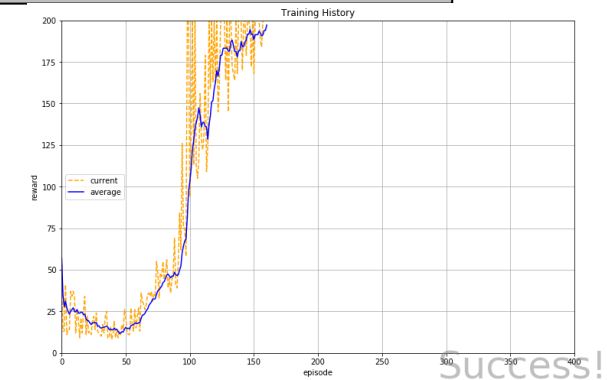
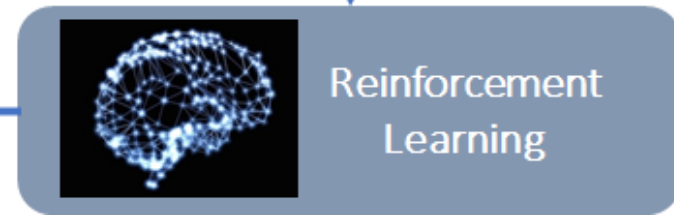
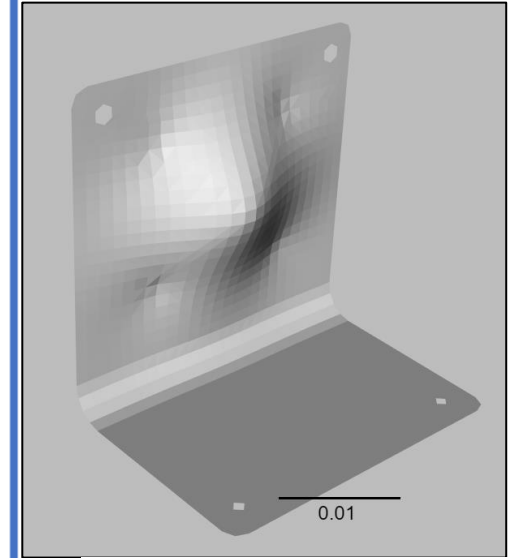


py [ Ansys ]

morph solve



receive a reward





# / Spray Dryer WebApp



# Use Ansys Python Manager – Graphical Installation (windows)

## Challenge/Need:

- **Anaconda GUI installer:** Transition to a **Commercial Business Model** was a major roadblock for Ansys Customers.
- Reduce Impact for Beginner & Intermediate Python users.

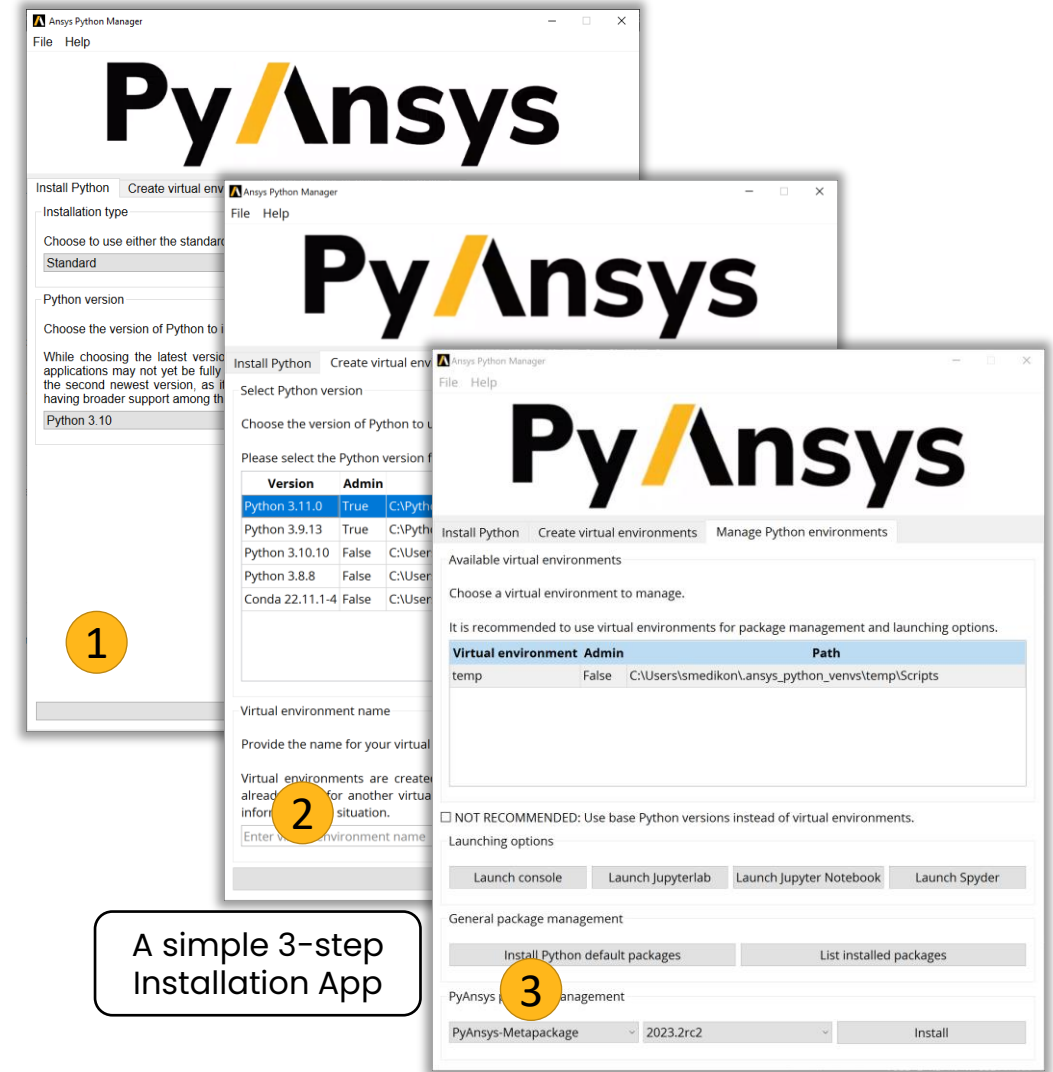
## Solution/Tool Created:

- A simple 3-step installer app created in <1 month.

- Using Ansys Python Manager:  
Download it from:

[Releases · ansys/python-installer-qt-gui \(github.com\)](https://github.com/ansys/python-installer-qt-gui)

<https://developer.ansys.com/ansys-python-manager>



# How to get started with PyAnsys (local installation)?

## 3 Step Process:

1. Install Ansys
2. Install Python and an IDE of your choice (Spyder, Jupyter, Jupyter Lab, ...)
3. Install PyAnsys modules

Release version	Release date	Click for more
Python 3.9.0	Oct. 5, 2020	<a href="#">Download</a> <a href="#">Release Notes</a>
Python 3.8.6	Sept. 24, 2020	<a href="#">Download</a> <a href="#">Release Notes</a>
Python 3.5.10	Sept. 5, 2020	<a href="#">Download</a> <a href="#">Release Notes</a>

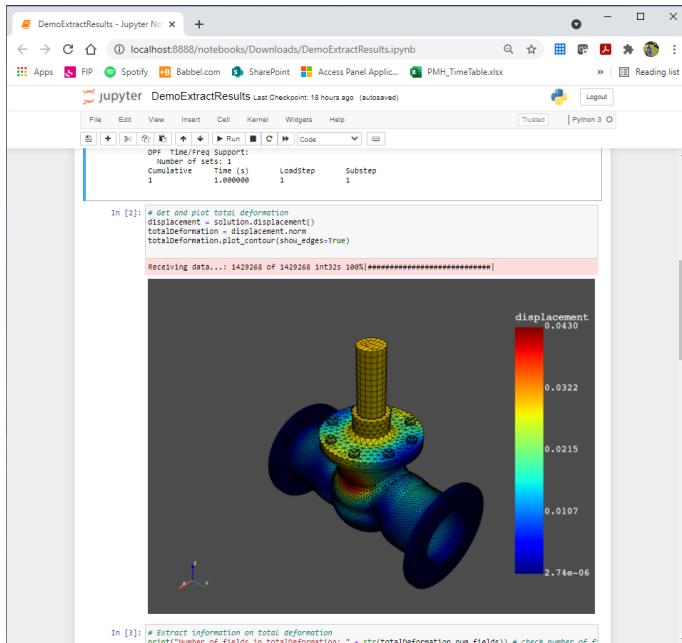
```
python -m venv PyAnsys
call PyAnsys\Scripts\activate
pip install jupyterlab
pip install ansys-mapdl-core
pip install ansys-mapdl-reader
pip install pyaedt
pip install ansys-dpf-core
pip install ansys-dpf-post
pip install ansys-fluent-core
pip install ansys-fluent-parametric
pip install ansys-fluent-visualization
```

[python.org/downloads/](https://python.org/downloads/)

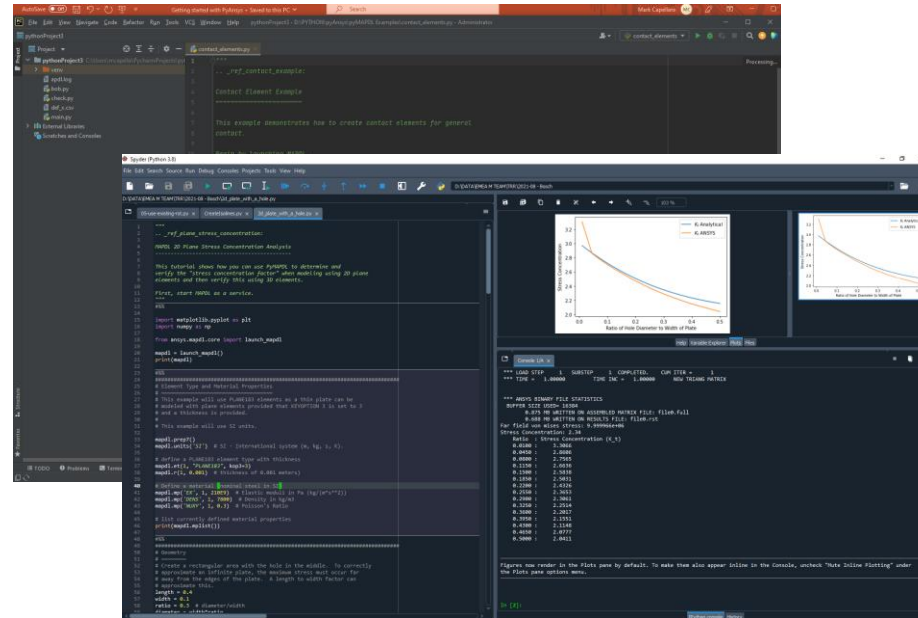
# What's the UI like?

## No specific UI. User's choice.

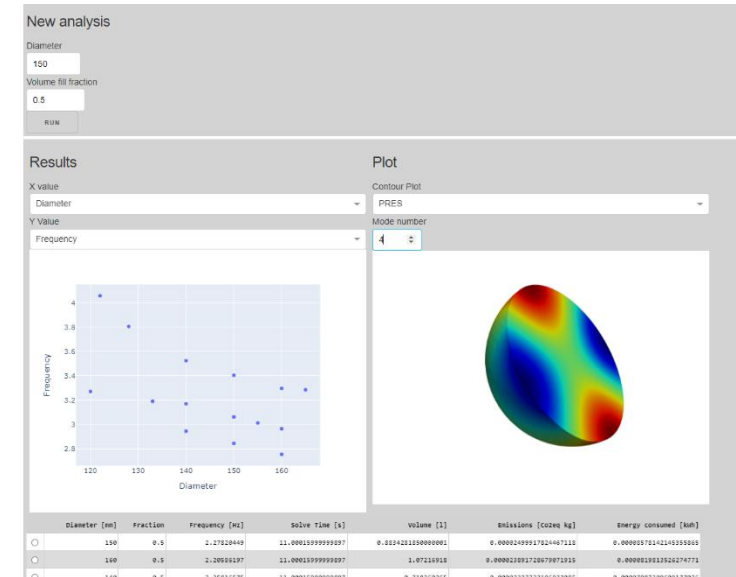
- PyAnsys is a set of tools to interact with Ansys products from a Python interface
- Any Python interface can be used (even the Windows Command Prompt !)
- Using an IDE (Integrated Development Environment) is recommended to write and maintain code.  
Many IDEs are available: [Spyder](#), [PyCharm](#),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



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



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

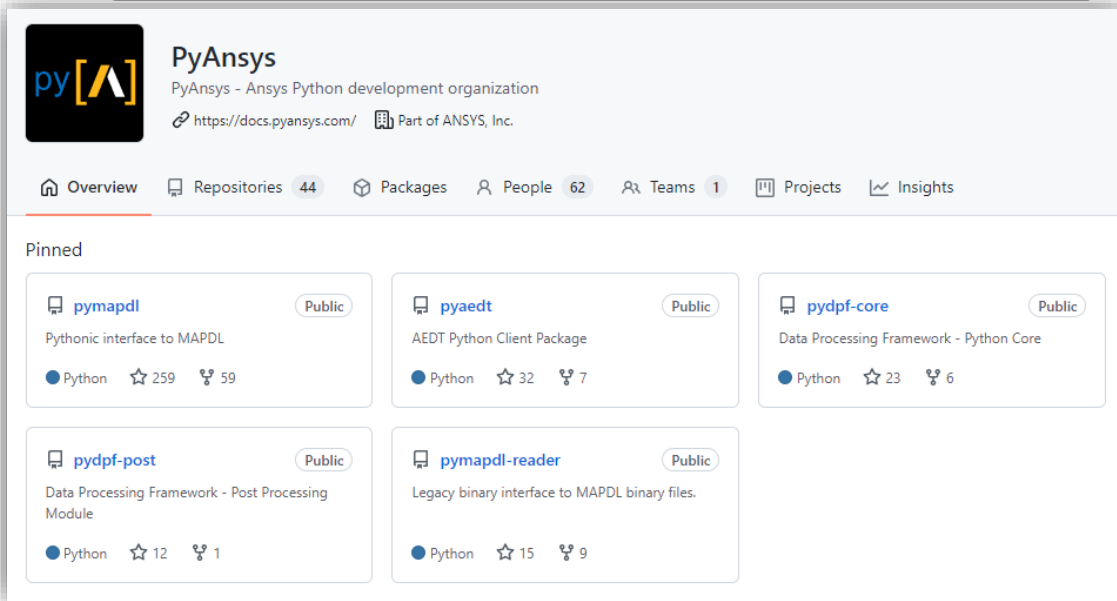


User can also develop apps by coding both the front end and the back end

# Documentation & Help?

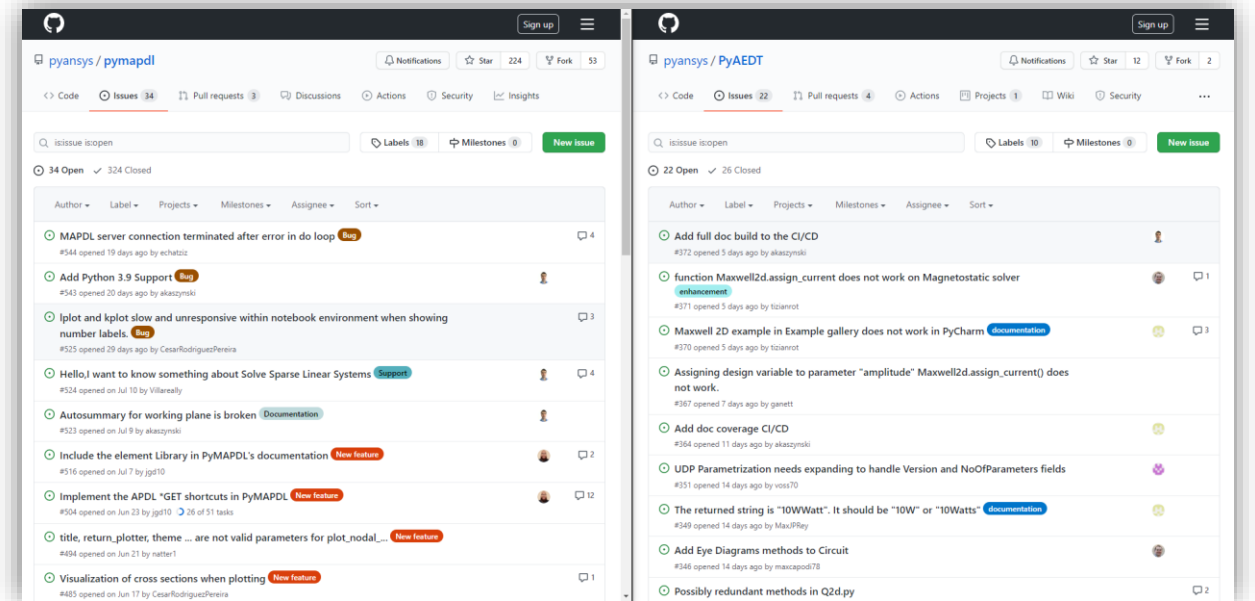
Source Code: <https://github.com/ansys/pyansys>

Contribute/Post-Issues directly on Github:



The screenshot shows the GitHub organization page for PyAnsys. The header includes the PyAnsys logo and the text "PyAnsys - Ansys Python development organization". Below the header, there are navigation tabs for Overview, Repositories (44), Packages, People (62), Teams (1), Projects, and Insights. The "Pinned" section displays several repositories:

- pymapdl** (Public): Pythonic interface to MAPDL. 259 stars, 59 forks.
- pyaedt** (Public): AEDT Python Client Package. 32 stars, 7 forks.
- pydpf-core** (Public): Data Processing Framework - Python Core. 23 stars, 6 forks.
- pydpf-post** (Public): Data Processing Framework - Post Processing Module. 12 stars, 1 fork.
- pymapdl-reader** (Public): Legacy binary interface to MAPDL binary files. 15 stars, 9 forks.

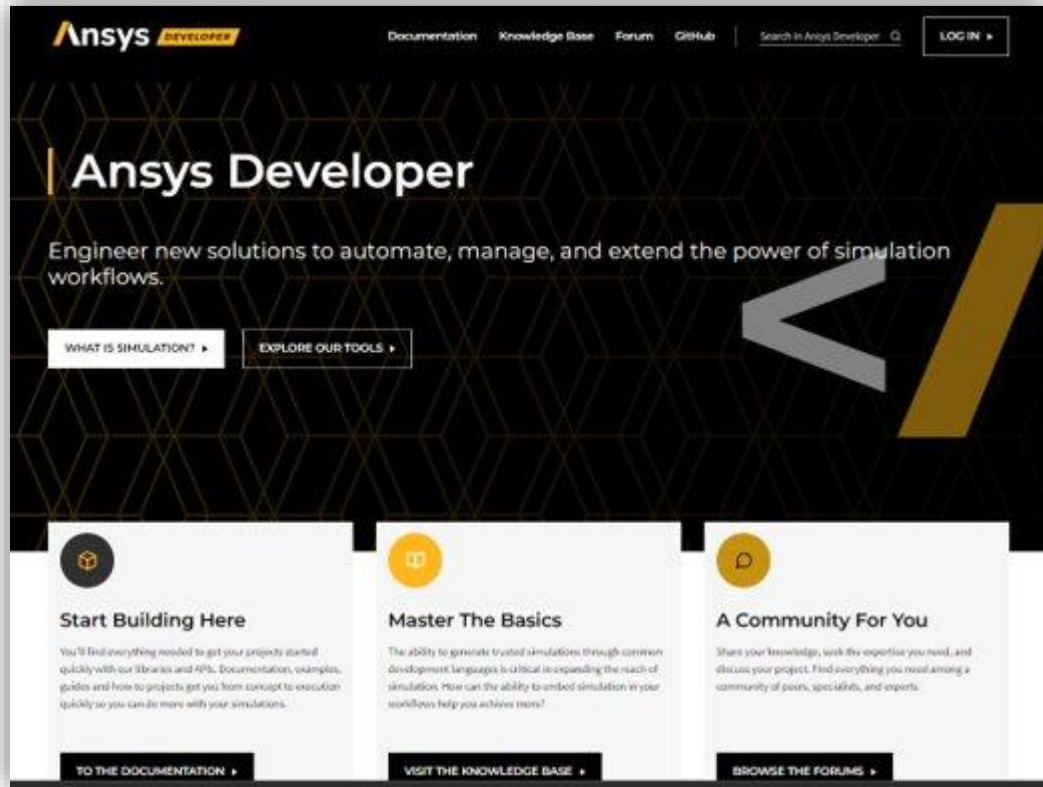


The screenshot shows two GitHub issue trackers side-by-side. The left tracker is for the `pymapdl` repository, showing 34 open issues. The right tracker is for the `PyAEDT` repository, showing 22 open issues. Both trackers list various issues with their titles, authors, and dates.

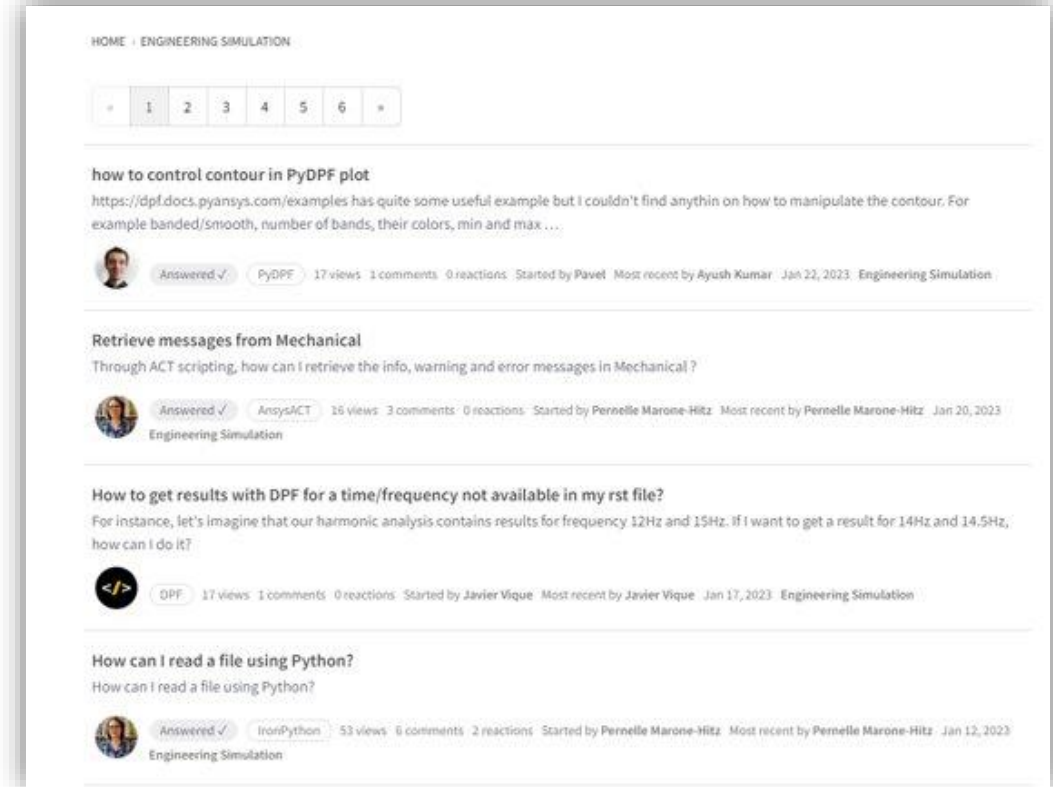
## Documentation:

- Access from GitHub or use these Direct links:
  - <https://docs.pyansys.com/>
  - <https://mapdl.docs.pyansys.com/>
  - <https://dpf.docs.pyansys.com/>
  - <https://aedt.docs.pyansys.com/>

# Need additional help? Use the Ansys Developer Portal and Developer Forum



Ansys Developer Portal:  
<https://developer.ansys.com/>



Ansys Developer Forum:  
<https://discuss.ansys.com/>



# Trainings (AIC & ALH)



Getting Started with PyMAPDL

Getting Started With PyMAPDL - Course Overview

PYTHON

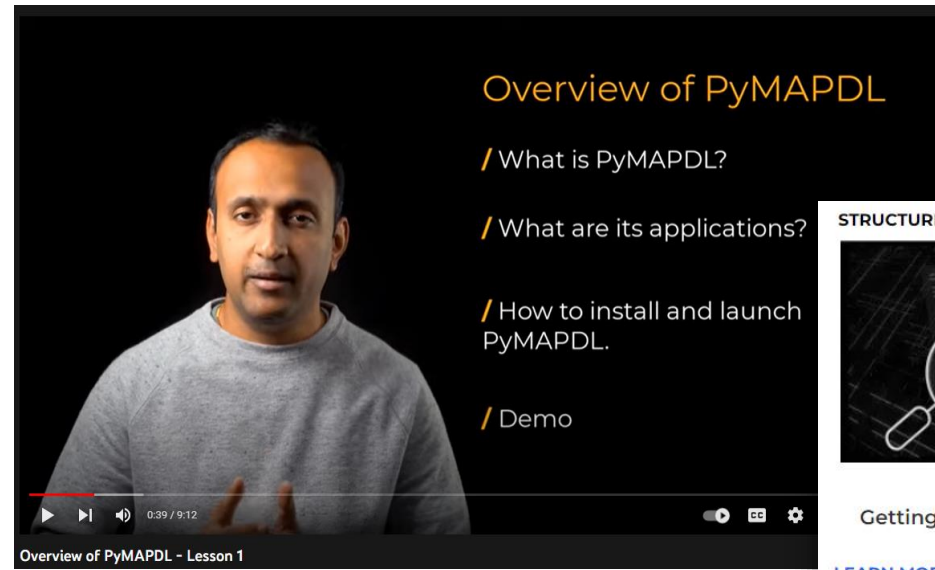


**New!**

LEARN SIMULATION

Intro to Python

LEARN MORE →




Overview of PyMAPDL

- / What is PyMAPDL?
- / What are its applications?
- / How to install and launch PyMAPDL.
- / Demo

Overview of PyMAPDL - Lesson 1

STRUCTURES



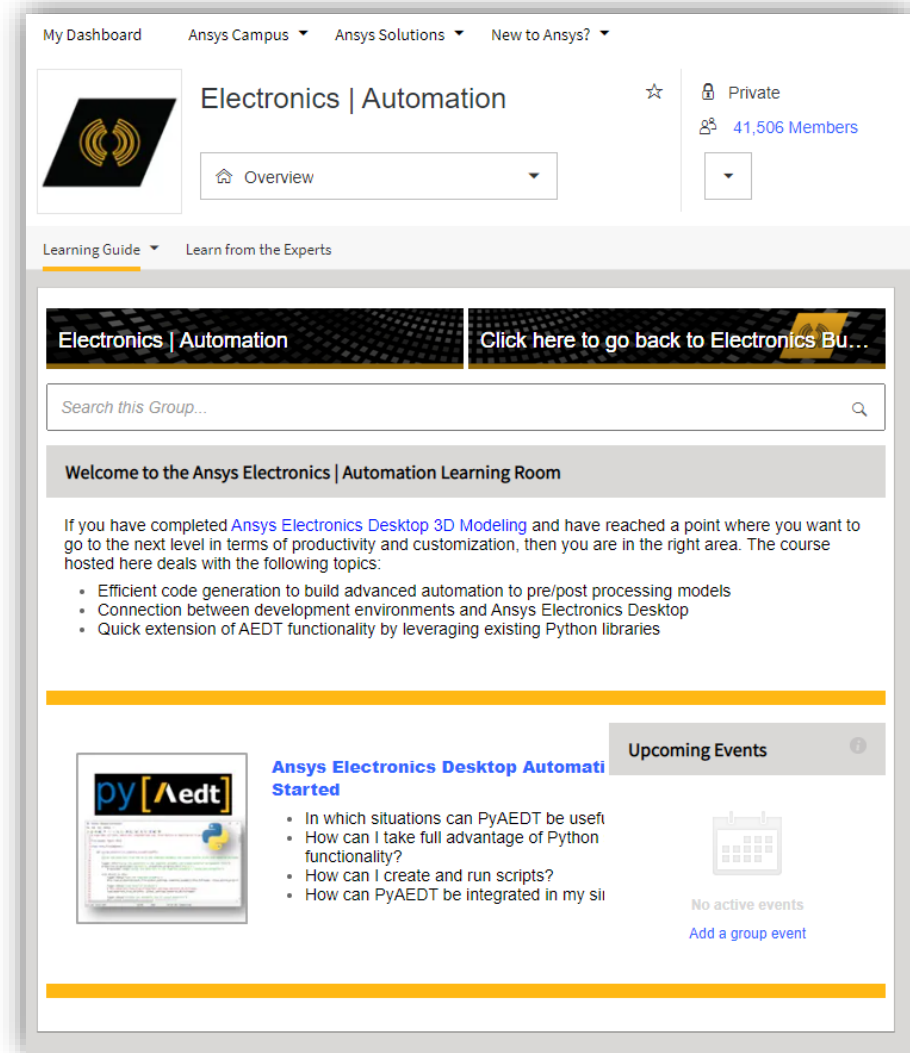
**New!**

LEARN SIMULATION

Getting Started with PyMAPDL

LEARN MORE →

**Ansys Innovation Courses (AIC)**



My Dashboard   Ansys Campus   Ansys Solutions   New to Ansys?

## Electronics | Automation

Private   41,506 Members

Overview

Learning Guide   Learn from the Experts

### Electronics | Automation

Click here to go back to Electronics Bu...

Search this Group...

Welcome to the Ansys Electronics | Automation Learning Room

If you have completed [Ansys Electronics Desktop 3D Modeling](#) and have reached a point where you want to go to the next level in terms of productivity and customization, then you are in the right area. The course hosted here deals with the following topics:

- Efficient code generation to build advanced automation to pre/post processing models
- Connection between development environments and Ansys Electronics Desktop
- Quick extension of AEDT functionality by leveraging existing Python libraries

#### Upcoming Events

**Ansys Electronics Desktop Automati Started**

- In which situations can PyAEDT be useft
- How can I take full advantage of Python functionality?
- How can I create and run scripts?
- How can PyAEDT be integrated in my sii

No active events

Add a group event

**Ansys Learning Hub (ALH)**

# PyAnsys: Module Cheat Sheets

## Cheat sheet for PyMAPDL

### Launching PyMAPDL

To launch PyMAPDL instance locally and exit it

```
from ansys.mapdl.core import LaunchMapdl
mapdl = LaunchMapdl()
mapdl.exit()
```

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

```
jobname = "user_jobname"
path = "path_of_directory"
LaunchMapdl(jobname=jobname, num_processors=4, working_directory=path)
```

To connect to an existing instance of MAPDL at IP 192.168.1.30 and port 5000.

```
mapdl = LaunchMapdl(ip="192.168.1.30", port=5000)
```

To create and exit a pool of instances

```
from ansys.mapdl.core import LocalMapdlPool
pool = LocalMapdlPool(10)
pool.exit()
```

### PyMAPDL Language

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

```
mapdl.esel('1', 'type', 'mesh')
```

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

```
mapdl.prep() # /PREP
mapdl.get() # /GET
```

In cases where removing '/' or will cause conflict with other commands, a prefix 'ansys/' or 'star/' is added.

```
mapdl.solu() # /SOLU
mapdl.slasholu() # ansys.solu()
mapdl.vget() # /VGET
mapdl.starvget() # star.vget
```

Getting Started with PyMAPDL / PyMAPDL on GitHub / Visit mapdltools.pyansys.com

## PyAEDT EDB-API Cheatsheet

### Launching EDB-API using PyAEDT

EDB manager manages the AEDB Database. An AEDB Database is a folder that contains the database representing any part of a PCB. It can be opened and edited using the EDB class.

```
edb = EdbManager()
edb.open('path_to_edb')
```

### Simulation Configuration

These classes are the containers of simulation configuration constructors for the EDB.

```
sim = SimulationConfiguration()
sim.add_simulation_configuration('sim', '10000')
sim.add_simulation_configuration('sim', '10000')
```

### Simulation Setup

These classes are the containers of setup classes in EDB for both HFSS and Siwave.

```
sim.setup.add_simulation_configuration('sim', '10000')
sim.setup.add_simulation_configuration('sim', '10000')
```

Getting Started with PyAEDT / Ansys Innovation Courses /

## PyAEDT-API Cheat sheet

### Launching PyAEDT

To launch HFSS instance locally and exit it

```
from ansys.aedt.core import LaunchPyAEDT
pyaedt = LaunchPyAEDT()
```

### Creating boundaries

In AEDT, open region is created as

```
box = pyaedt.create_box('box', [0, 0, 0], [1, 1, 1])
```

### Mesh Class

Mesh module manages the mesh functions in PyAEDT

```
mesh = Mesh(mesh_name='mesh', num_elements=10000)
```

### Port definitions

Common port types in HFSS are Lumped-port and Wave guide-port. Pythonic way of defining the lumped-port is

```
port = pyaedt.create_lumped_port('port', [0, 0, 0], [1, 1, 1])
```

### Analysis Class

The solution setup (setup) in HFSS design is analyzed by calling the following Python command

```
analysis = AnalysisClass()
analysis.solve()
```

### Post-processing Class

Post-processing class has modules for creating and editing plots in AEDT. They are accessible through the post library

```
post = PostProcessingClass()
post.create_plot('plot')
```

### Geometry Creation

If you need to create a geometry in HFSS, you can use the create\_box method. It is used to create a box in HFSS. A box is defined as follows

```
box = pyaedt.create_box('box', [0, 0, 0], [1, 1, 1])
```

### Setup Class

Setup class is used to define the solution setup in the PyAEDT

```
setup = SetupClass()
setup.create_simulation_configuration('sim', '10000')
```

Getting Started with PyAEDT / Ansys Innovation Courses /

## Cheat sheet for PyFluent

### Solver Settings Object Interface

### Launch Fluent locally

The following method is used to start Fluent from Python in GUI mode. This code starts Fluent in the background so that commands can be sent to Fluent from the Python interpreter

```
from ansys.fluent.core import launch_fluent
launch_fluent(mode='gui')
```

### Define materials

This example shows you how you use Solver settings objects to define materials

```
material = Material(name='steel', density=7800, elastic_modulus=200000000000.0)
```

### Define boundary conditions

The examples in this section show how you use Solver settings objects to define boundary conditions.

```
inlet = VelocityInlet(name='inlet', velocity=[1, 1, 1])
```

### Apply solution settings

PyFluent allows you to use Solver settings objects to apply solution settings, initialize and solve

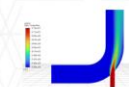
```
solver = SolverSettings()
solver.solve()
```

### Post-Processing

PyFluent allows you to post-process data with results object. The following examples show you how to create and display contours on a plane

```
contour = Contour(name='contour', field='velocity')
```

### Temperature Contour



### Import mesh in launch session

The following examples show how to read the available mesh file in Fluent Session.

```
mesh = Mesh(name='mesh', file_name='mesh.msh')
```

### Enable heat transfer physics

The following examples show how to enable heat transfer by activating the energy equation.

```
energy = EnergyEquation()
energy.enable()
```

### References from PyAnsys Documentation

- Getting Started
- PyFluent Solver Settings Objects
- PyFluent Examples

Getting Started with PyAnsys / PyAnsys on GitHub / Visit dev.docs.pyansys.com

## PyDynamicReporting Cheatsheet

### Version 0.4.0 (stable)

### Anyans Dynamic Reporting Service

To launch and stop a local ADR service on a given database

```
from ansys.dynamicreporting import AnyansDynamicReportingService
service = AnyansDynamicReportingService('localhost:5000')
```

### Set plot properties

To display a table as a plot

```
plot = Plot(name='plot', data=[1, 2, 3])
```

### Tag Items

To set tags on an item

```
item = Item(name='item', tag='tag')
```

### Create new items

To create new items in the database of different types

```
item = Item(name='item', value=1)
```

### Visualize items and reports

To search for text items

```
search = Search(name='search', text='text')
```

### Get URL for items and reports

To get the URL for the first item returned

```
url = Url(name='url', item='item')
```

### References from PyAnsys Documentation

- Getting Started
- API reference
- Examples


Getting Started with PyDynamicReporting / PyDynamic on GitHub / Visit dynamicreporting.docs.pyansys.com

And many more on....

[cheatsheets.docs.pyansys.com](https://cheatsheets.docs.pyansys.com)

or

[developer.ansys.com](https://developer.ansys.com)



# Testimonials – Research Community

Pyansys tremendously helped by providing an **easy** way to procedurally read and process data directly from Ansys **without any human intervention**. Combined with Python's enormous data analysis and **plotting capability** this helped me automate extraction of knowledge from 100s of giant Ansys simulations and saved a lot of expensive human hours

Ph.D. Student

You can **easily** retrieve node stresses and temperatures. Then outside, there is a material database which you can retrieve yield strength based on temperatures, and finally you can calculate factor of safety. **And perhaps you can plot it**. Think about you have many analyses in different locations and you can automatize this process

Aerospace Engineer

I was trying to avoid exporting my results to csv and then importing back into python. [I] started to look also into the "controller" features, which I came to like a lot.

I also contributed to pyansys once, to support reading transient analyses with a reduced number of DOFs [...]. As far as I remember it was written in C, which I don't "speak".

Ph.D. Student

Quite **easy** to embed the simulation in different environments. I also have to say, how glad I am that the plotting functionality in pyansys is so much faster and intuitive than its APDL counterpart

Undergraduate Student

data extraction

data transforms

plotting

automation

solver embedding

user contributions





# Conclusions

- PyAnsys introduces a **paradigm shift** in how Ansys simulation tools will be used going forward. Ansys is the '**first**' simulation software provider to introduce such **dynamic interaction** with its products.
- PyAnsys has been deployed on **GitHub** and is **controlled by ANSYS**.
- Ability to separate **Pre-processing, FE model** and **Post-processing** from outside the Ansys environment is a strength that PyAnsys has and will help us **deploy, maintain** and **scale** for applications across various industries.
- **AI/ML Community** can easily integrate ANSYS physics capabilities into their processes.



The Ansys logo consists of a yellow slanted bar followed by the word "Ansys" in a bold, black, sans-serif font.

