Getting Started with Analysis Automation
using Ansys Mechanical Scripting

Nov. 17th, 2023

Contents

©2022 ANSYS, Inc.

Toolization and automation Workflow

Ansys

Image of development flow

1. Organize and document what you want to implement

   Automation is intended to be used by someone other than the author. By documenting and discussing the purpose, content, and form of use, you can create better tools.

2. Consider whether existing functions and ACT can be utilized

   If you are having trouble making a decision, please contact Technical Support.

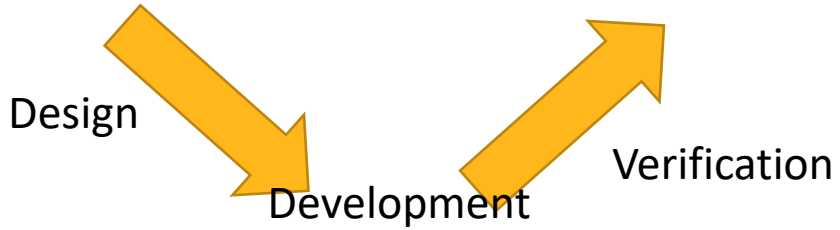
3. Decide how you want users to use what you want to do
4. Break down what you want to do into detailed tasks
5. Write code for each task
6. Operation verification (debugging)

7. Check if the tool is as expected
8. Documentation maintenance and provision to users

   Document maintenance is always necessary for maintenance and handover.

Supported by Mechanical features

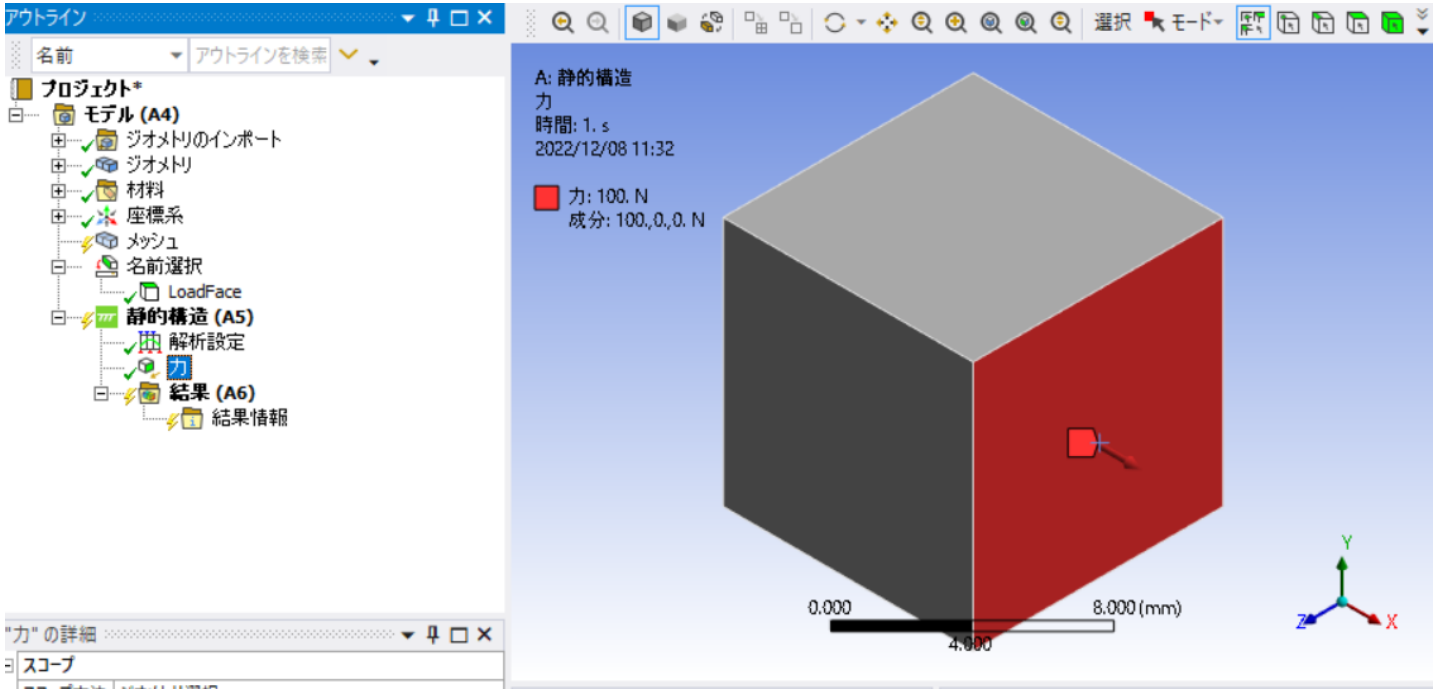
The so-called "V-shaped model"



©2022 ANSYS, Inc.

Specifically... Organize your requirements

- I will introduce a specific example of creating a simple sample.
  - Objective: Set a load on the face specified by name selection
  - If complex processing is required, it is recommended to create a flowchart.



©2022 ANSYS, Inc.

Deployment/Usage

How users use Python scripts

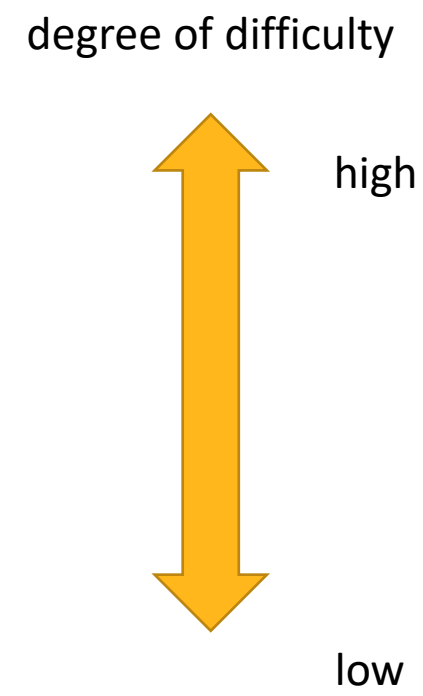
There are various methods as follows, so you need to choose the appropriate one.

- Build and distribute to WBEX format

- Set as a script object in Mechanical
  - Python Code Objects
  - Python result

- Add Python scripts to Mechanical GUI
  - User button

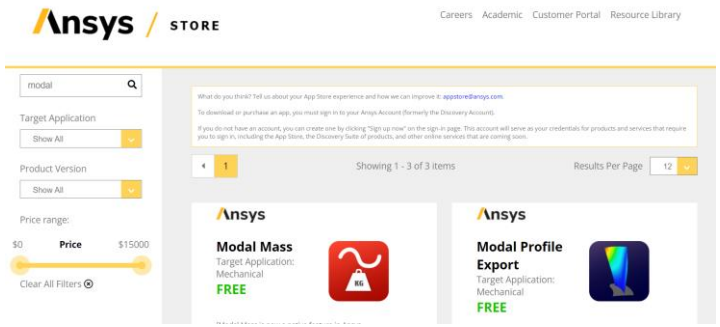
- Other: Mechanical script execution from Workbench

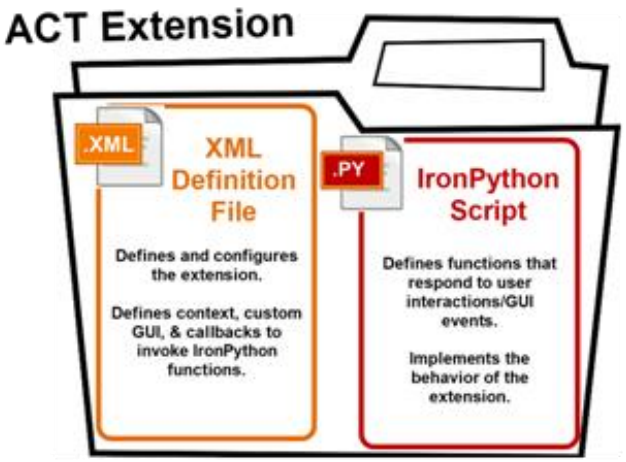


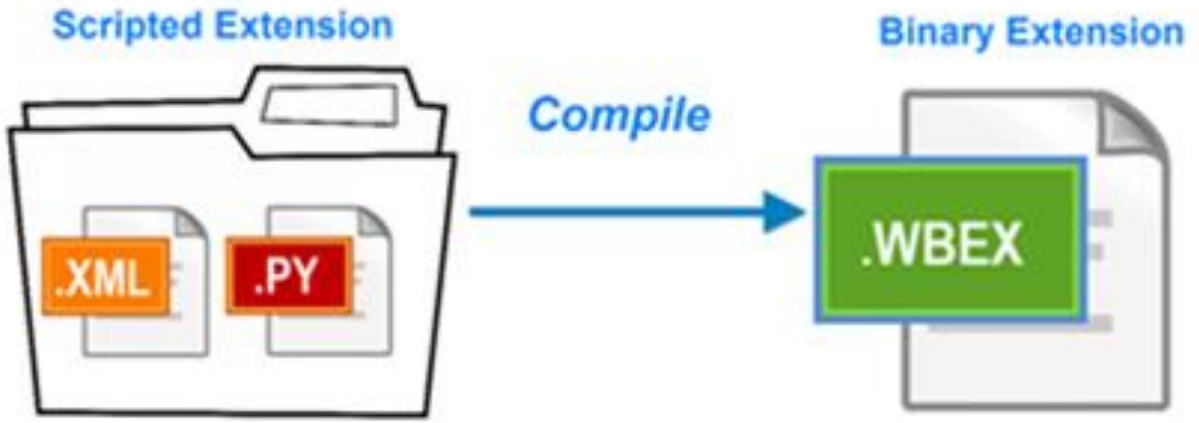
©2022 ANSYS, Inc.

ACT (wbex form)

- Added extensions and wizards (ability to navigate instructions)
- The Ansys App Store distributes the ACT in this format.
- Enterprise license required for debug compilation (build)
- Not only Python but also XML etc. are required
  - Python: Describes the processing to be performed according to the user's operation
  - XML: Describes the buttons, display contents and corresponding Python on the Mechanical GUI







©2022 ANSYS, Inc.

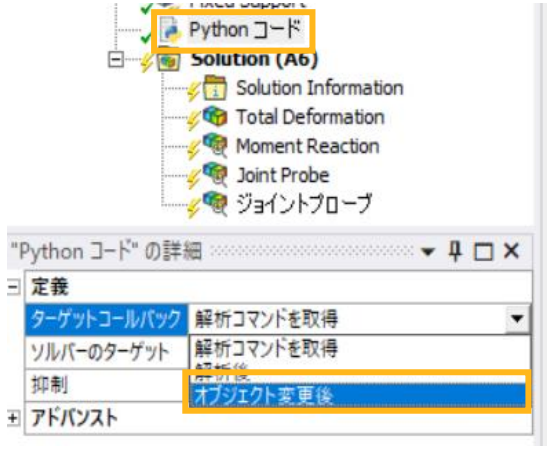
Python Code Objects

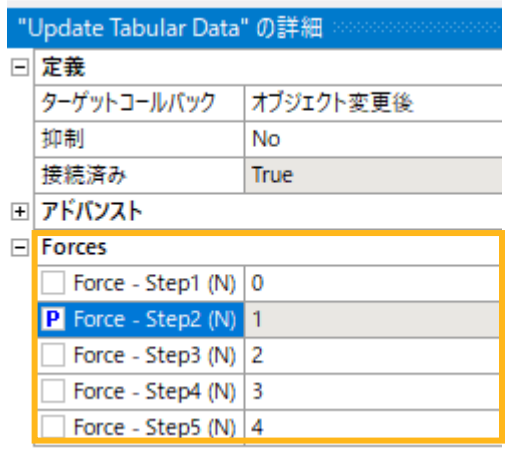
- There are two ways to use it.
  - Use the Python API to get model information and insert MAPDL commands accordingly
  - Execute Python code based on events that occur in Mechanical (configuration changes, analysis start and end)

  - Property providers can be configured to allow users to add input fields

  →Items that cannot be parameterized can be analyzed as parameters (table load, pretension, etc.)

- Reference Help
  - Help:Chapter 18, Using Python Code(ansys.com)
  - Examples of using Python code objects(ansys.com)

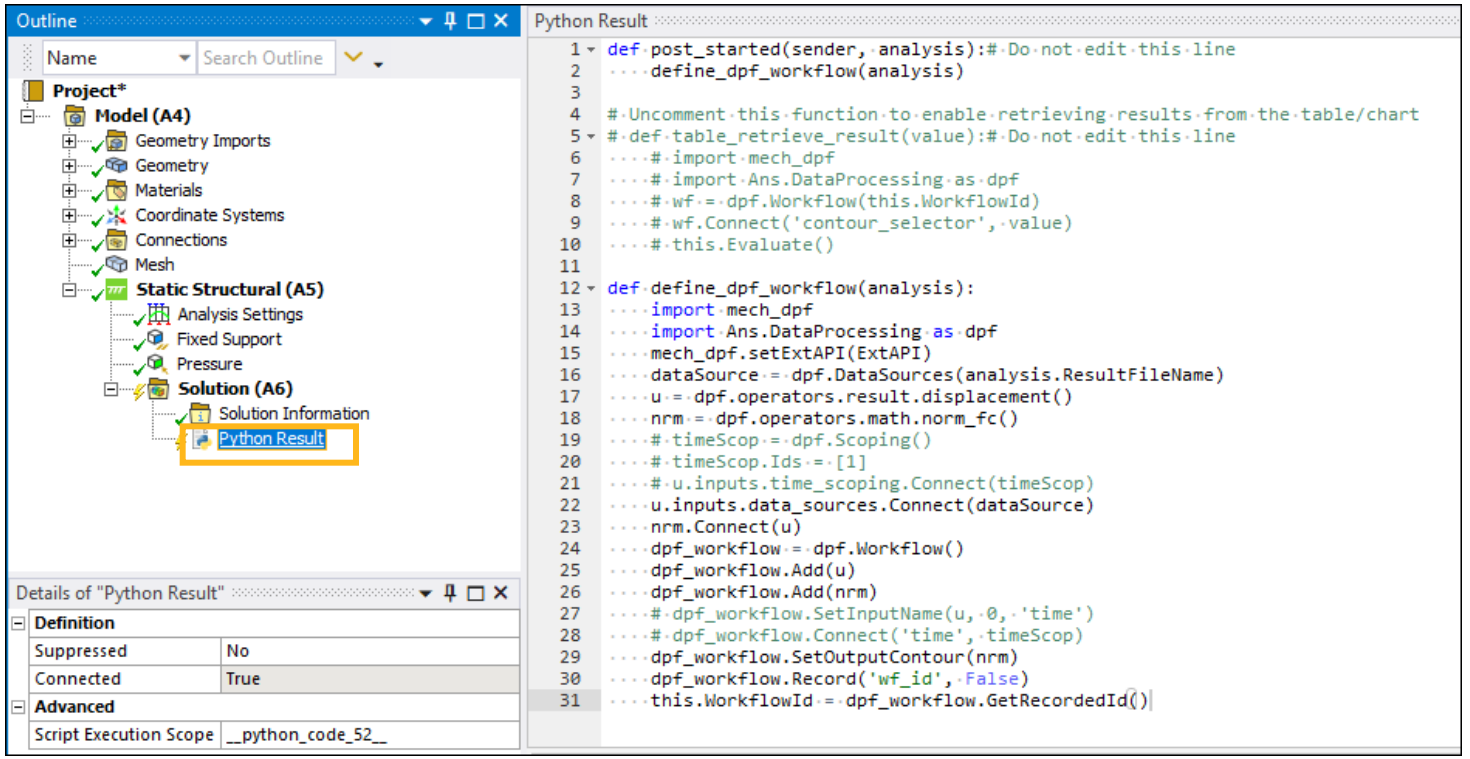




©2022 ANSYS, Inc.

Python results

- Use the Data Processing Framework to create customized post-processing
- By combining operators, various data processing is possible
- ALH Lectures: Structure | Customization - Getting Started with Ansys Data Processing Framework(sapjam.com)



16.16. Python results (ansys.com)

©2022 ANSYS, Inc.

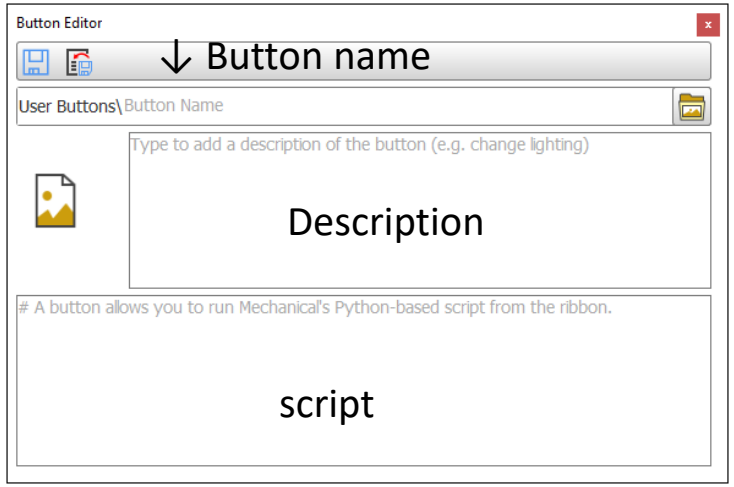
What is a user button?

- Useful when you want to register Python scripts as buttons on the toolbar → turn frequent operations into tools.
- Python scripts can be imported by themselves→ easier to set up and distribute than creating an ACT

How to create

- On the Automation tab, the [User Button] [Manage] → button editor is displayed.
- Register with an icon, name, description, and script
- From the toolbar of the editor, user buttons can also be registered.

Icon→



Creating a User-Defined Button(ansys.com)

©2022 ANSYS, Inc.

Working window

The Mechanical Console

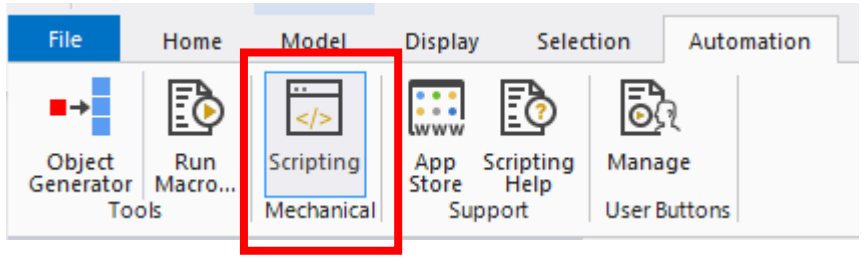
To start the console:

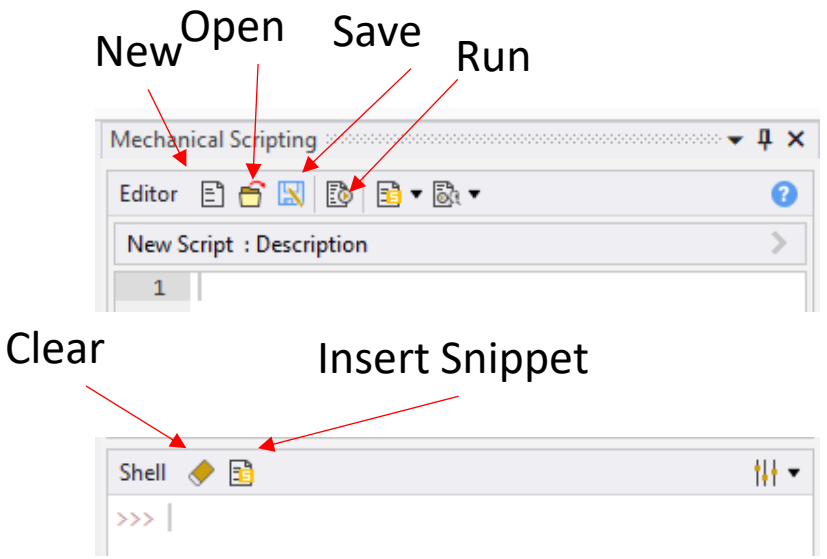
- Open mechanical 2020R1 or later
- Under Automation select Scripting

The console will appear on the right of the screen

To know what each button does simply hover on it and the definition will appear

- The commands written in the script editor (top) will be executed when pressing run (or ctrl+F5)
- Any script written in the script editor can be saved and reopen in a new session
- The commands written in the shell (bottom) will be executed when pressing enter
- Anything written in the shell can be cleared
- A good way to get started is to use ‘Insert Snippet’; which has a list of basic API functions

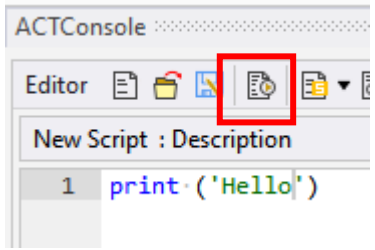


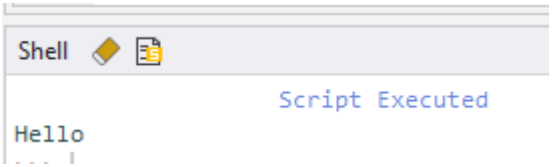


©2022 ANSYS, Inc.

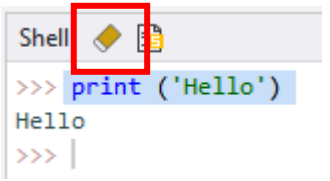
Your first script

- Open the console
- In the script editor write print (‘Hello’)
- Hit ‘Run’ or Ctrl+F5
- Look at the Shell window





- Clear the Shell window and write print (‘Hello’) in this window
- Hit Enter



As we have just seen either the Shell or the editor can be used to write scripts

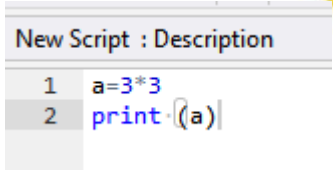
©2022 ANSYS, Inc.

A word about python

- The mechanical console uses python as a programing language
- More training about python can be found online or in the ACT training materials

We are going to try some basic python functionalities, by using the script editor:

- In the script editor write ‘a=3*3’ on the first line and ‘print (a)’ on the second line
- Run the script

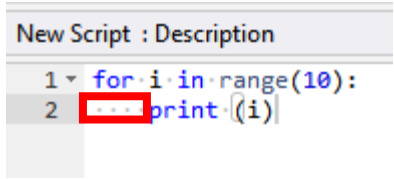


The simplest loop in python is the for loop. The for loop needs to be indented

- In the script editor write:

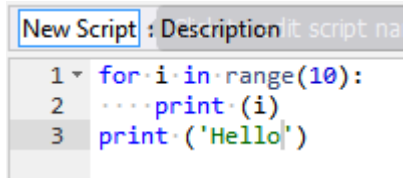
```
for i in range(10):
    print (I)
```

And run

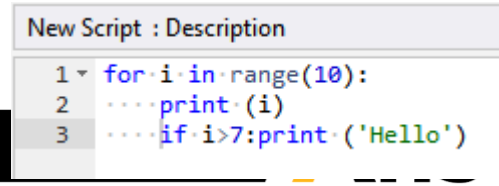


Pay attention to the space in front of print. Anything written with an indent under for will be inside the loop. To stop the loop, just remove the indent

To try it: under print (i), write print (‘Hello’) with an indent and without. Run each version
Look at the shell window each time



In the same way we can create more indents for a ‘if’ statement for example



Note here range(10) goes from 0 to 9

©2022 ANSYS, Inc.

Task for Decomposition API concept understanding

Ansys

Why do you need to understand APIs?

- Once you have decided on an understanding of what you want to do (flowchart), break it down into executable tasks
- How granular should I break down the task on the Mechanical side? ?

- **Mechanical API is object-oriented**
  - Change and manipulate (create, duplicate) the values of objects to build an analysis model
  - Task decomposition in units of objects (≒ each item in the analysis tree) is basic.

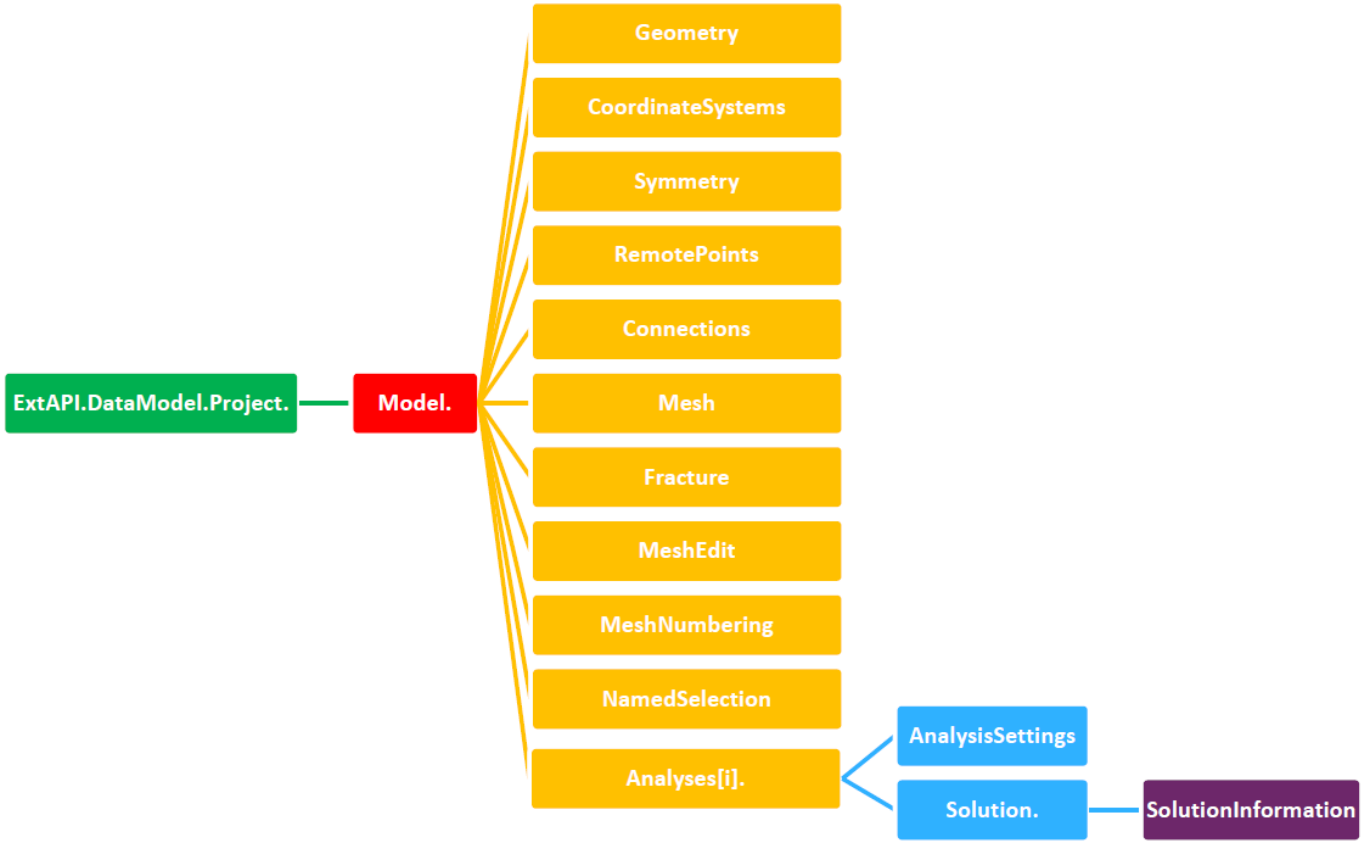
→Explain the object-oriented concepts required for Mechanical scripting
  - Object Attributes: Methods and Properties
  - How to get and manipulate objects

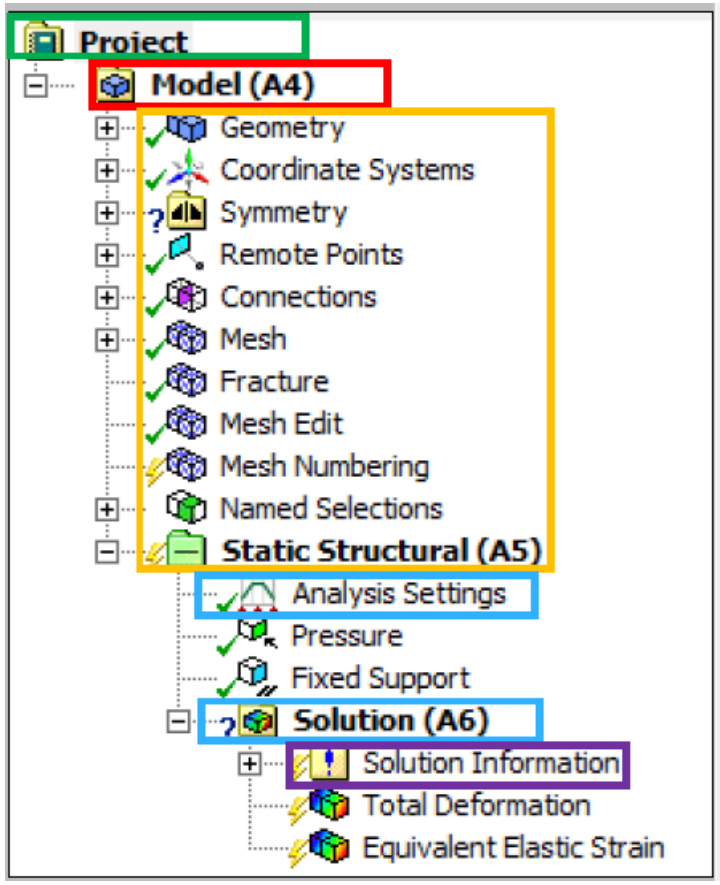
※There are other important object-oriented concepts (inheritance, polymorphism, etc.), but in automation, you don't need to be particularly aware of them.

©2022 ANSYS, Inc.

Mechanical Automation API

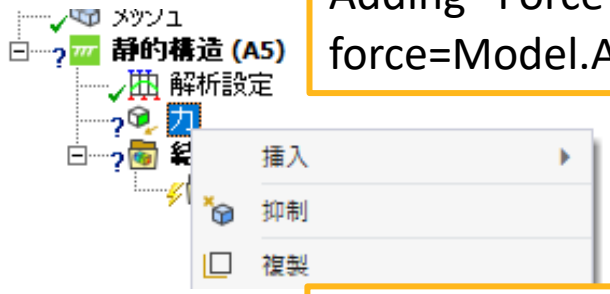
```
# Reference model
model = ExtAPI.DataModel.Project.Model
```





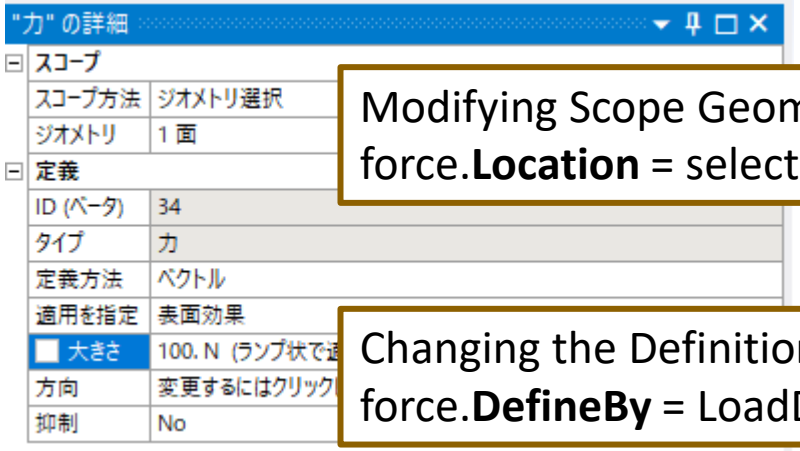
©2022 ANSYS, Inc.

Object Attributes Methods and Properties



Adding “Force" :
force=Model.Analyses[0]**.AddForce()**

Duplication of “Force" :
force2 = force.**Duplicate()**



Modifying Scope Geometry
force.**Location** = selection

Changing the Definition Method
force.**DefineBy** = LoadDefineBy.Components

In Mechanical, the image is as follows.

- **Method** = Right-click menu
  - Creation and duplication of objects, etc.
  - Use as a function
- **Property** = Detail Items
  - Change advanced settings
  - Basically assign and change

※Some do not fit the above classification.

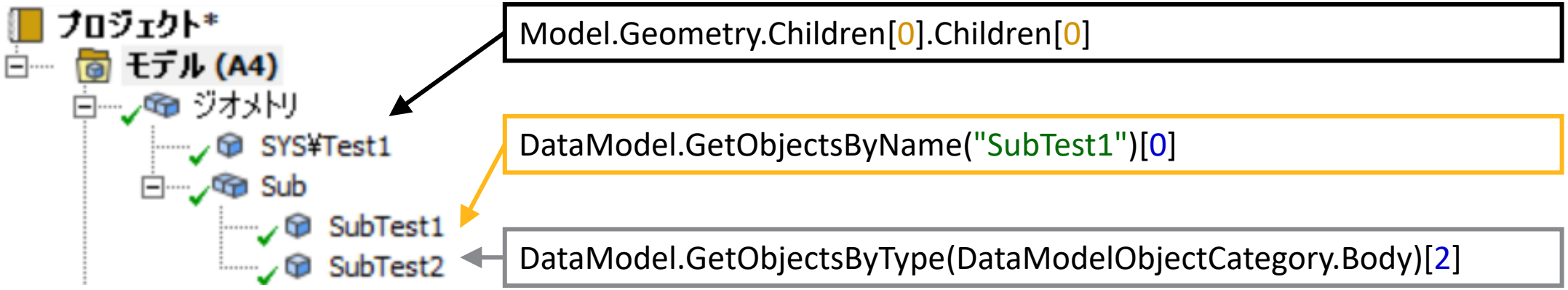
Please understand that it is only a guide.

©2022 ANSYS, Inc.

Get an object How to access the parse tree

- Retrieve, manipulate, and modify objects→ It's important to understand how to get it！

- You can get an object in the following way:
  - **Location-based in a tree using Children**
  - **Name-based selection of objects**
  - **Batch selection of the same type**

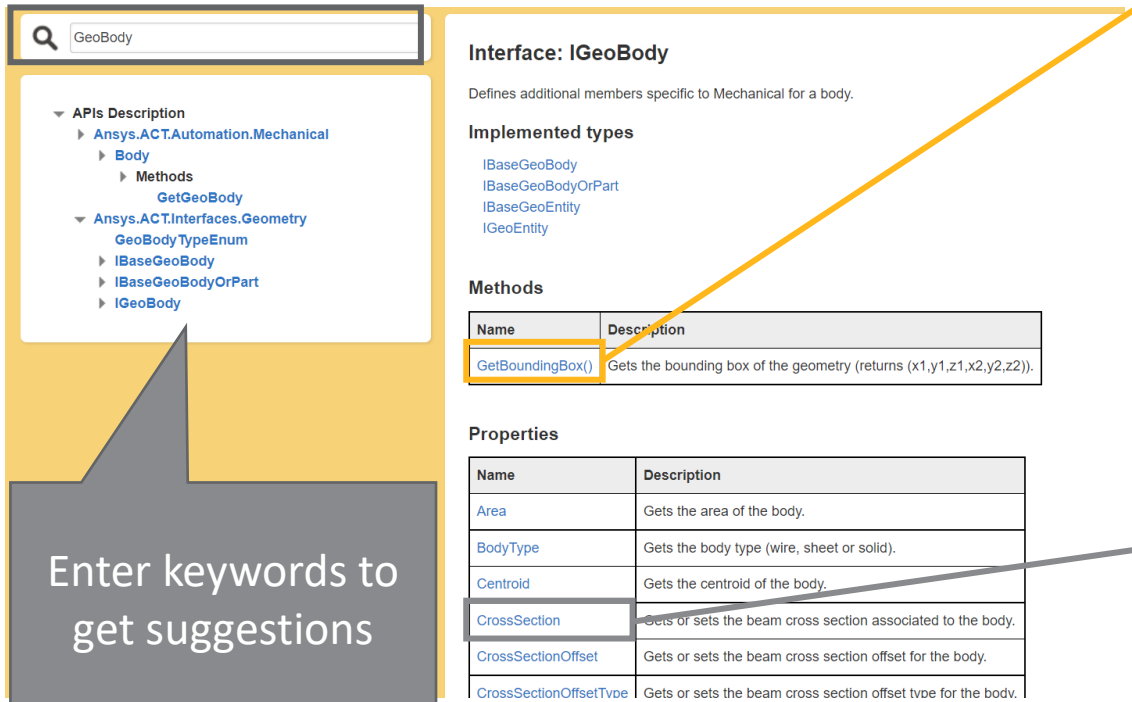


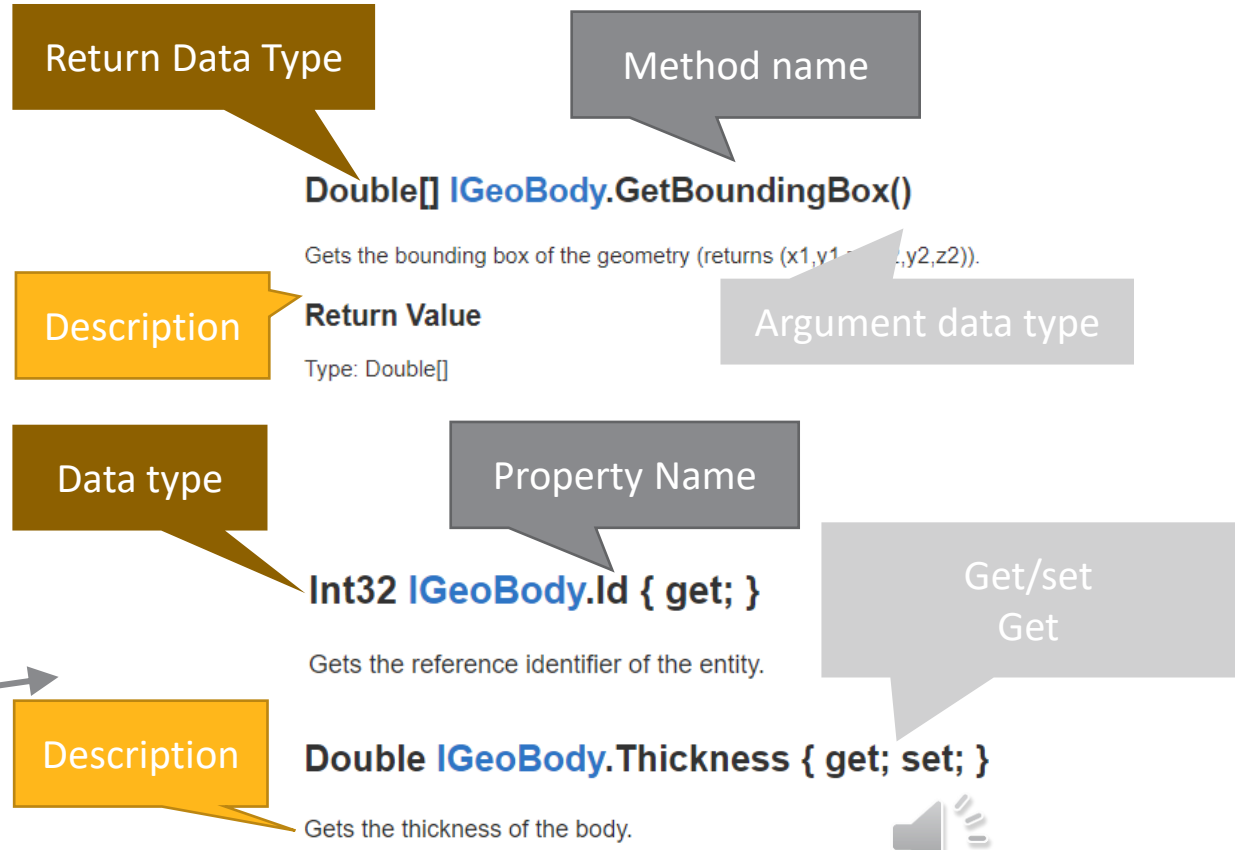
©2022 ANSYS, Inc.

How to read the API Reference for manipulating objects

- Once you can get the appropriate object, you need to know how to set it up.
  - API Reference: A description of object-specific operations and values
- If you search for a class interface,
- Methods and properties are listed。

https://storage.ansys.com/api-a/v222/ACTReferenceHTML/Mechanical/index.html

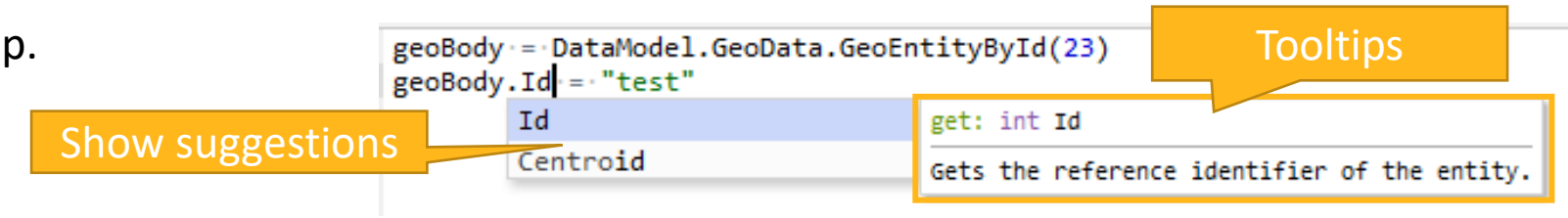




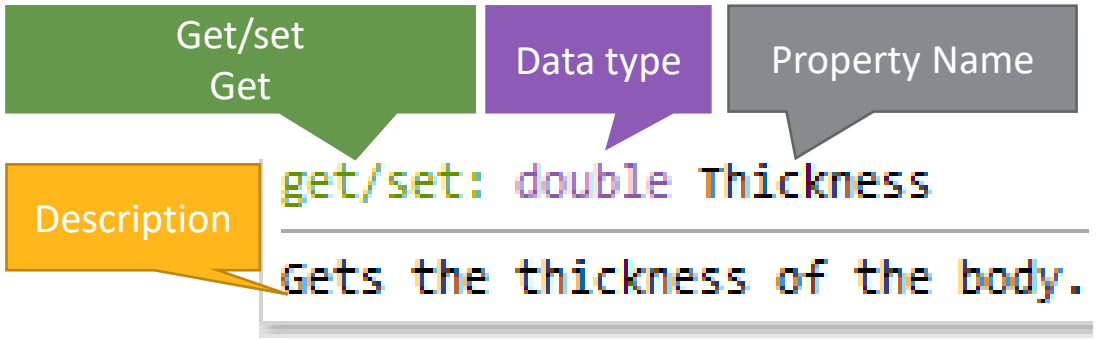
©2023 ANSYS, Inc. / Confidential

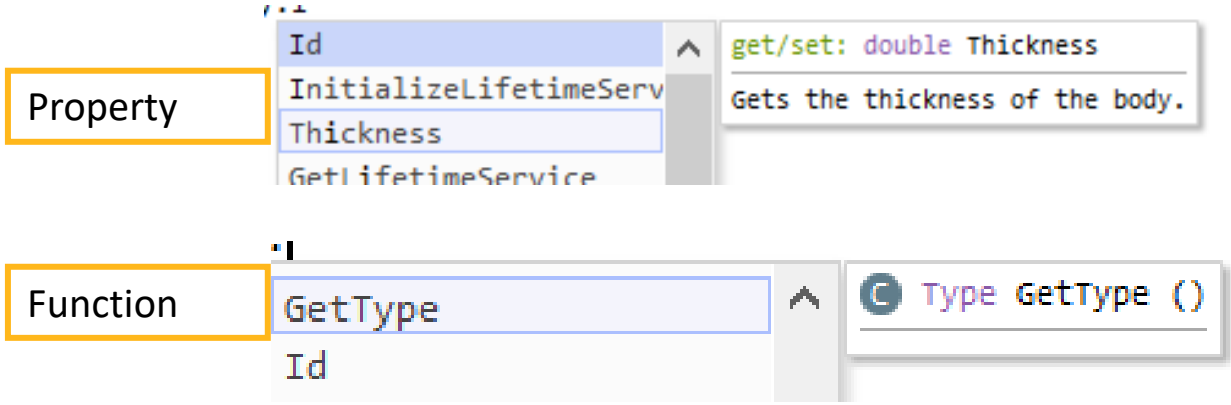
Work with objects Autocomplete

- It is also possible to examine the API as you write code.
  - Prediction to prevent spelling mistakes.
  - The tooltip also shows simple help.



- What the tooltip displays





※You can also check it by using the dir function (showing available attributes), help function (showing help) on the shell

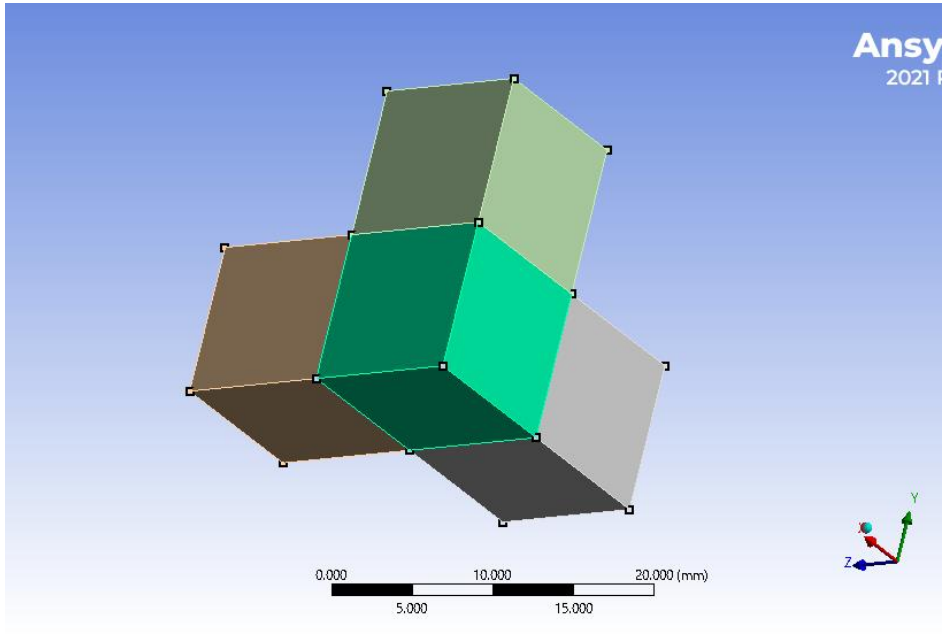
©2022 ANSYS, Inc.

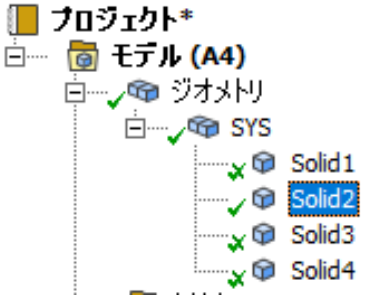
Note: Difference between tree body and geometry body

Record the operation for each and confirm the ID

→**A body on the tree and a body on the geometry have different IDs.！**

Care must be taken when handling IDs in code



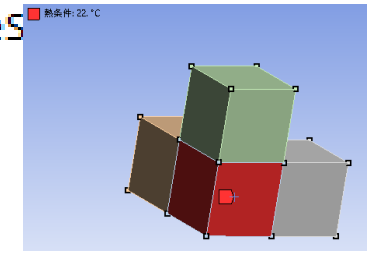


Changing Stiffness Behavior (Body on Tree)

```
#region Details View Action
body_20 = DataModel.GetObjectById(20)
body_20.StiffnessBehavior = StiffnessBehavior.Rigid
#endregion
```

Set as Thermal Condition Target (Body on Geometry)

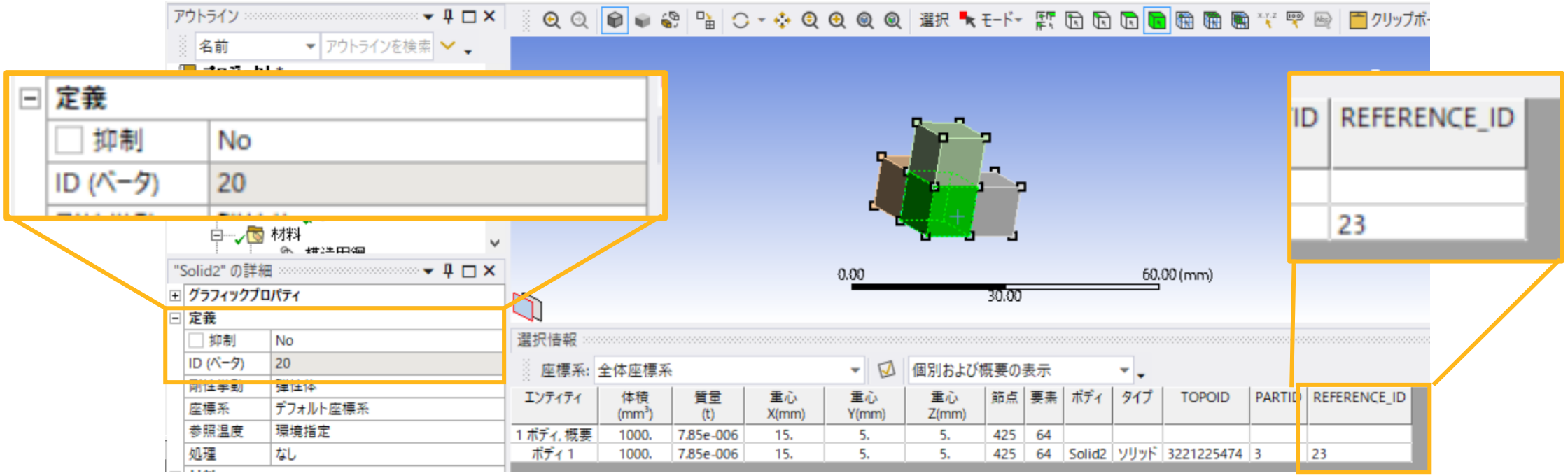
```
#region Details View Action
selection = ExtAPI.SelectionManager.CreateS
selection.Ids = [23]
thermal_condition_39.Location = selection
#endregion
```



©2022 ANSYS, Inc.

Note: How to find the body ID on the GUI

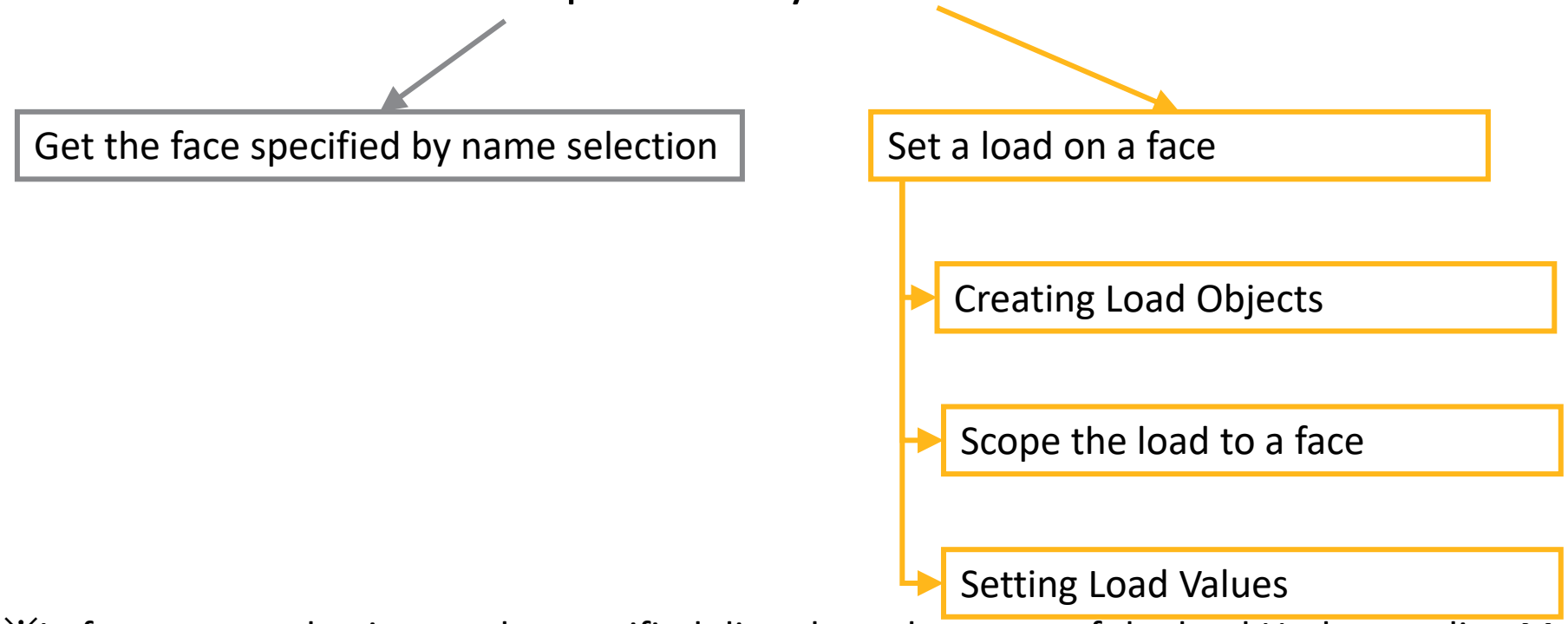
- You can check each of them from the following points on the GUI.
  - Tree: "ID (beta)" in "Details" (Note: beta option required)
  - Geometry Data: "REFERENCE_ID" of the Selection tool



©2022 ANSYS, Inc.

Specifically... Task decomposition

- Set a load on the face specified by name selection



※In fact, name selection can be specified directly to the scope of the load Understanding Mechanical is also an important point

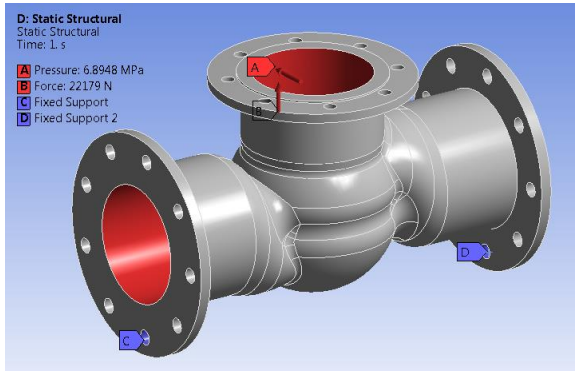
©2022 ANSYS, Inc.

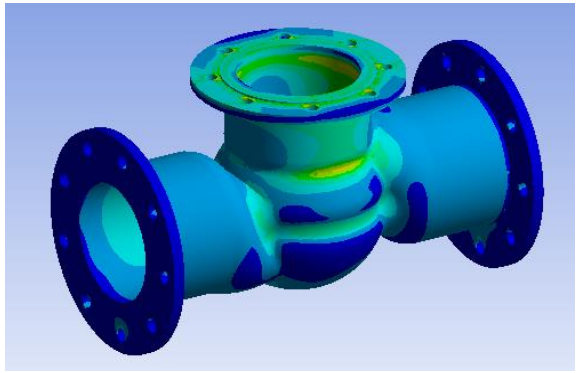
Hands on workshop

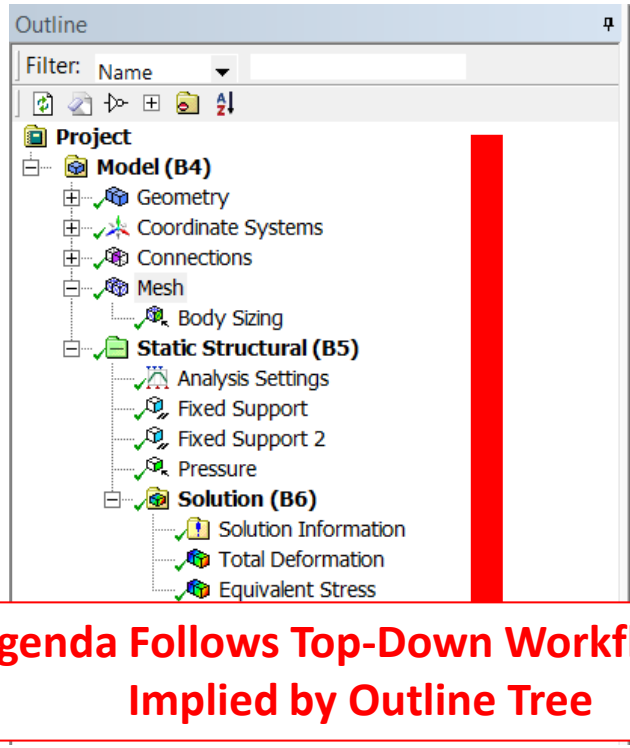
Ansys

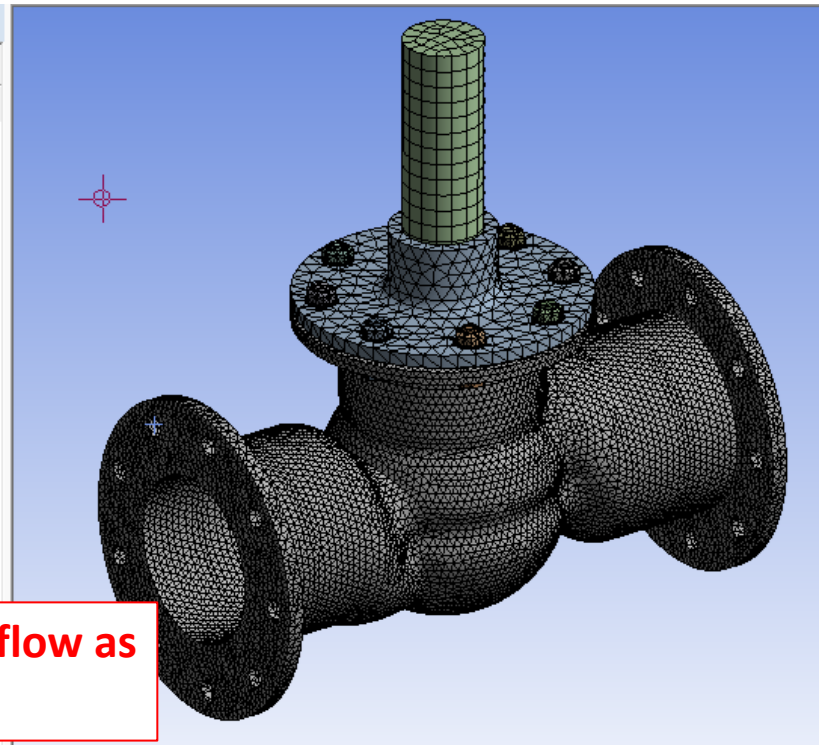
Tube example

- Module 01: Material assignment
- Module 02: Create NameSelection
- Module 03: Mesh Sizing
- Module 04: Boundary condition
- Module 05: Solver Settings
- Module 06: Create a Result
- Module 07: Extract result data









Agenda Follows Top-Down Workflow as Implied by Outline Tree

©2022 ANSYS, Inc.

Material assignment

- For material assignment, we need to use the skill of “body on tree”
  1. Find body on tree
  2. Loop all body to assign the material
  3. Create a filter to assign right material to each other

```
for Geo in Model.Geometry.GetChildren(DataModelObjectCategory.Body, True):
    #Geo=Assemble.Children[0]
    if 'ValveBody' in Geo.Name or 'flange' in Geo.Name:
        Geo.Material='Type 40 Gray Cast Iron'
    elif 'seal' in Geo.Name:
        Geo.Material='Type 302 Stainless Steel'
    else:
        Geo.Material='AISI 6150 Steel'
```


©2022 ANSYS, Inc.

Create NameSelection

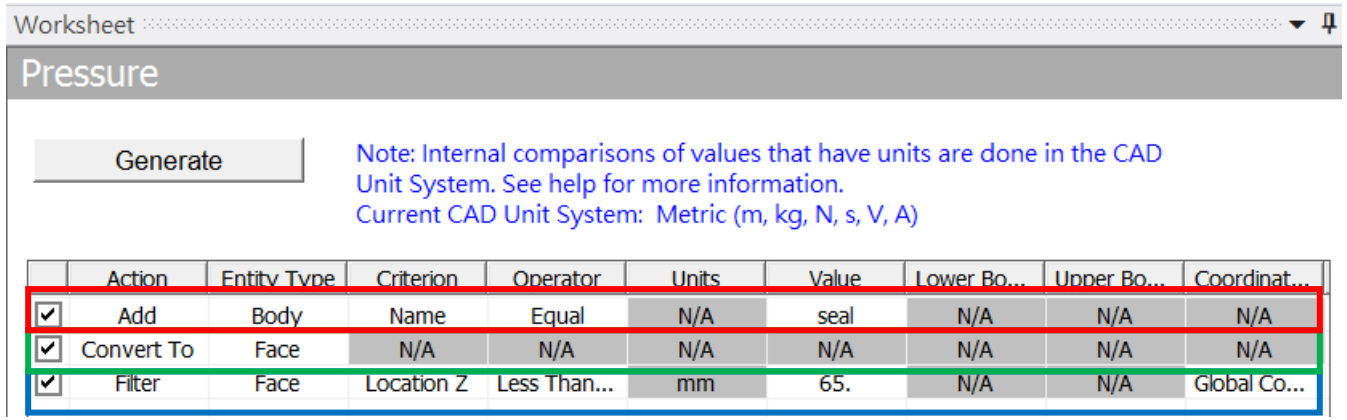
- In Mechanical preprocessing we do a lot of settings by apply the Geometry. But for Script is hard to let the code to “Click GUI”.
- NameSelection will be a very efficiency way to point which entity that will be used later.
- There is two ways to create NameSelection
  1. NameSelection Worksheet
  2. SelectionManager

©2022 ANSYS, Inc.

Create NameSelection by worksheet

- Basically, every action from worksheet can be rewrite by python code

```
#Pick buttom of Seal worksheet
Nsel=Model.AddNamedSelection()
Nsel.ScopingMethod=GeometryDefineByType.Worksheet
Nsel.Name="Pressure"
pressws=Nsel.GenerationCriteria
pressws.Add(None)
pressws[0].EntityType=SelectionType.GeoBody
pressws[0].Criterion=SelectionCriterionType.Name
pressws[0].Operator=SelectionOperatorType.Equal
pressws[0].Value="seal"
pressws.Add(None)
pressws[1].Action=SelectionActionType.Convert
pressws[1].EntityType=SelectionType.GeoFace
pressws.Add(None)
pressws[2].Action=SelectionActionType.Filter
pressws[2].EntityType=SelectionType.GeoFace
pressws[2].Criterion=SelectionCriterionType.LocationZ
pressws[2].Operator=SelectionOperatorType.LessThanOrEqual
pressws[2].Value=Quantity("65 [mm]")
Nsel.Generate()
```



©2022 ANSYS, Inc.

Create NameSelection by worksheet

- Another NameSelection

```
#Create Nameselection by worksheet
for i in range(2):
    NS=Model.AddNamedSelection()
    NS.ScopingMethod=GeometryDefineByType.Worksheet
    NS.Name="Fixed"+str(i)
    NSWS=NS.GenerationCriteria
    NSWS.Add(None)
    NSWS[0].EntityType=SelectionType.GeoFace
    NSWS[0].Criterion=SelectionCriterionType.LocationY
    if i==0:
        NSWS[0].Operator=SelectionOperatorType.Largest
    else:
        NSWS[0].Operator=SelectionOperatorType.Smallest
    NS.Generate()
```


©2022 ANSYS, Inc.

Create NameSelection by Selection Manager

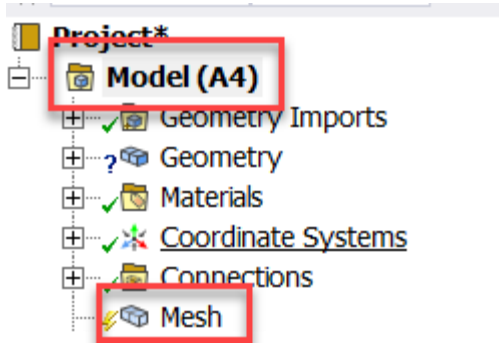
- If you what to use Selection Manager to create the NameSelection or apply the Geometry. There are two things you must to do
  1. Pick the GeoBody no matter what feature you want to pick.
  2. Use Selection manager to create a selection

```
#Pick buttom os seal by Geobody
SealBody=DataModel.GetObjectsByName("seal")
SealGeobody=SealBody[0].GetGeoBody()
Pressureface=[]
cenZ=1e9
for face in SealGeobody.Faces:
    if face.Centroid[2]<cenZ:
        cenZ=face.Centroid[2]
        Pressureface=face
    print(cenZ)
#Selection Manager
selmgr=ExtAPI.SelectionManager
selmgr.ClearSelection()
Sealsel=selmgr.CreateSelectionInfo(SelectionTypeEnum.GeometryEntities)
Sealsel.Entities=[Pressureface]
NS=Model.NamedSelections.AddNamedSelection()
NS.Location=Sealsel
NS.Name="Preesure2"
```


©2022 ANSYS, Inc.

Mesh Sizing

- Let’s use Selection Manager to define Mesh Sizing directly




```
#Mesh sizing
selmgr.ClearSelection()
sel=selmgr.CreateSelectionInfo(SelectionTypeEnum.GeometryEntities)
for Geo in Model.Geometry.GetChildren(DataModelObjectCategory.Body, True):
    if Geo.Name in ['bolt','nut1']:
        sel.Entities=[Geo.GetGeoBody()]
        selmgr.AddSelection(sel)

mesh=Model.Mesh
mesh.ElementSize=Quantity("5 [mm]")
meshsizing=mesh.AddSizing()
meshsizing.ElementSize=Quantity("2 [mm]")
selmgr.ClearSelection()
mesh.GenerateMesh()
```


©2022 ANSYS, Inc.

Boundary condition

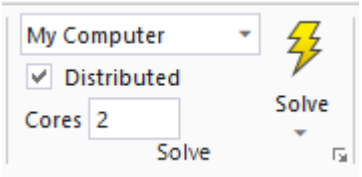
- Again, follow the same logic to assign boundary condition, but this time let’s use NameSelection to Scope the Geometry.

```
#·Boundary·settings
Analysis=Model.Analyses[0]
for·NS·in·Model.NamedSelections.Children:
····if·(NS.Name).Contains('Fixed'):
········Fixedsupport=Analysis.AddFixedSupport()
········Fixedsupport.Location=NS

Pressure=Analysis.AddPressure()
Pressure.Location=Sealsel
Pressure.Magnitude.Output.SetDiscreteValue(0,Quantity("20·[MPa]"))
```


©2022 ANSYS, Inc.

Solver Settings

You can access the Mechanical solver settings and change how an analysis is being solved
To do so use:
ExtAPI.Application.SolveConfigurations["My Computer"]
Or
ExtAPI.Application.SolveConfigurations["My Computer, Background"]



For example to get the license in use:
config=ExtAPI.Application.SolveConfigurations["My Computer"]
print config.Settings.License

```
1  config=ExtAPI.Application.SolveConfigurations["My Computer"]
2  print config.Settings.License
```

```
1  analysis = Model.Analyses[0]
2  analysis.Solve(1)
```

The number of core settings can be controlled by using:
config.SolveProcessSettings.MaxNumberOfCores=4

Note. The number of core displayed in the toolbar will not change

©2022 ANSYS, Inc.

Solver Settings

The analysis settings can be changed by using:

analysis.AnalysisSettings…

For example:

analysis.AnalysisSettings.NumberOfSteps=2

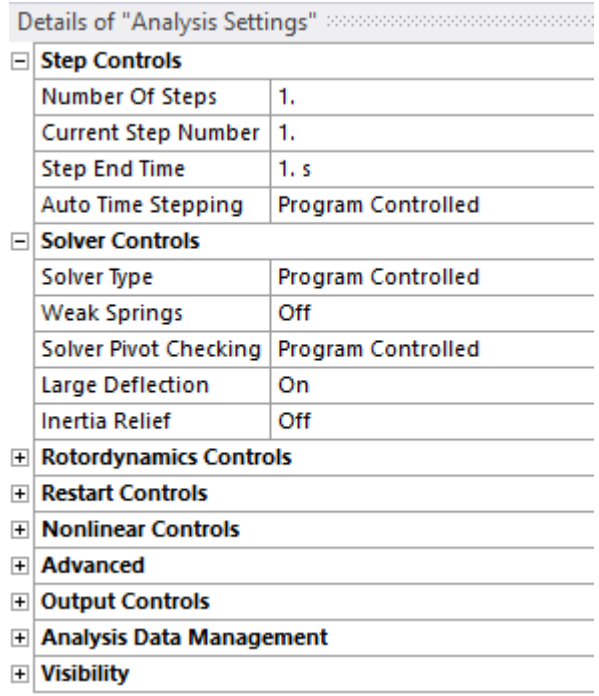
An analysis can be solved by using

analysis = Model.Analyses[0]

analysis.Solve(True)

And can be cleared by using:

analysis.ClearGeneratedData()



©2022 ANSYS, Inc.

Create a Result

You can create any result object by using analysis.Solution. The available results will be listed:

***analysis.Solution.AddTotalDeformation()* will create a deformation result**

Once one or several result object have been created, they can be evaluated by using

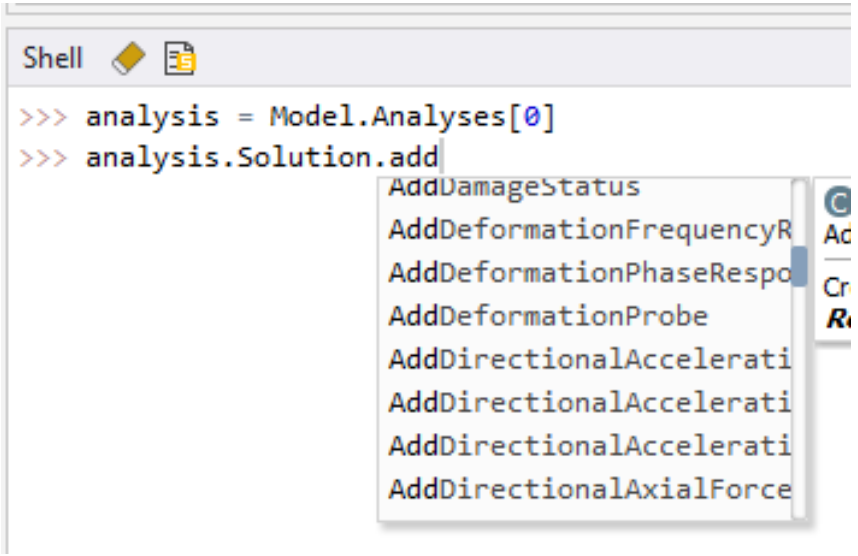
analysis.solve(True) or result.EvaluateAllResults()

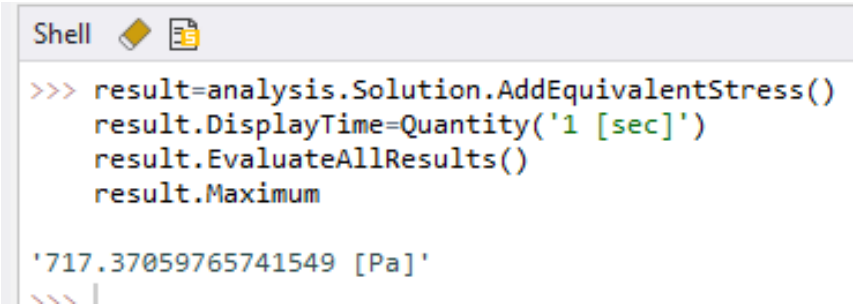
The scoping of an object is the same as the scoping of a load, described on module 2

The detail of a result object may be changed for example

```
result=analysis.Solution.AddEquivalentStress()
result.DisplayTime=Quantity('1 [sec]')
result.EvaluateAllResults()
result.Maximum
```

Will return the maximum equivalent stress





©2022 ANSYS, Inc.

Extract result data

The values for a result can be extracted using ***result.Plotdata***
For example:

```
result=analysis.Solution.AddEquivalentElasticStrain()
result.EvaluateAllResults()
result.PlotData
```

```
import csv
fpath=analysis.WorkingDir+result.Name+'.csv'
with open(fpath, mode='wb') as rfile:
    wr = csv.writer( rfile)
    wr.writerows(zip(*[result.PlotData['Node'],result.PlotData['Values']]))

```

Or if the results is already evaluated:

```
result=Tree.ActiveObjects[0]#will select the result selected in the tree
result.PlotData
```

The plot data always contain both the node/element number and the values
The values available are listed under:

```
result.PlotData.Keys
```

result.PlotData['key'] will create a list of the values under this key. For example to get the first value:
Print (result.PlotData['Values'][0]) or the first node number ***print (result.PlotData['Node'][0])***

Let's assume we want to write the values to a text file in the solver directory:
import csv

```
fpath=analysis.WorkingDir+result.Name+'.csv'
```

with open(fpath, mode='wb') as rfile:

```
wr = csv.writer( rfile)
wr.writerows(zip(*[result.PlotData['Node'],result.PlotData['Values']]))
```

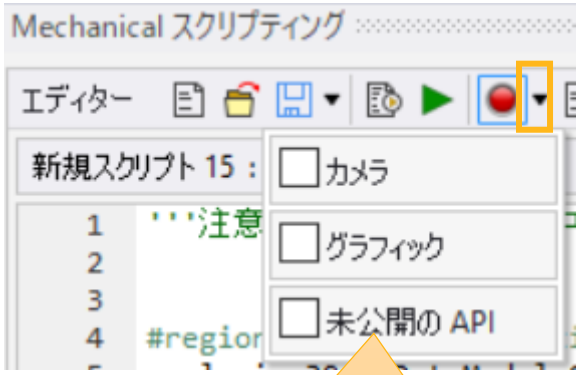

©2022 ANSYS, Inc.

Efficient code creation support

Ansys

Code Creation Support API Recording

- Automatically generate (log) supported APIs based on actions performed
- **Because the recording feature is under development not all workflows are recorded.**



If you open the drop-down list from ▼ on the right side of the button, you can change the setting to record camera operations, etc.

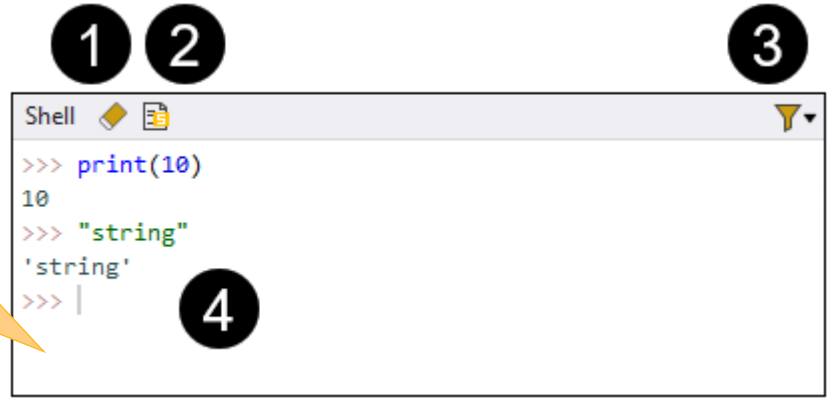
©2022 ANSYS, Inc.

Code Creation Shell

- At the bottom of the editor, enter the Python code and press the [Enter] key to execute it immediately.
- Useful for creating and testing short code snippets
- Develop smoothly by experimenting with small test code and creating longer code

- Also recommended when trying out the Scripting Guide sample (introduced later)
  - It's hard to understand if you do everything at once, so do it in groups of several lines

If you want to break a line in the shell (for statement, etc.), **enter [Shift]+[Enter].**
you can use **1**arrow to select a previously executed command。



1. Clear content
2. Insert snippet
3. Shell settings
4. Command input location

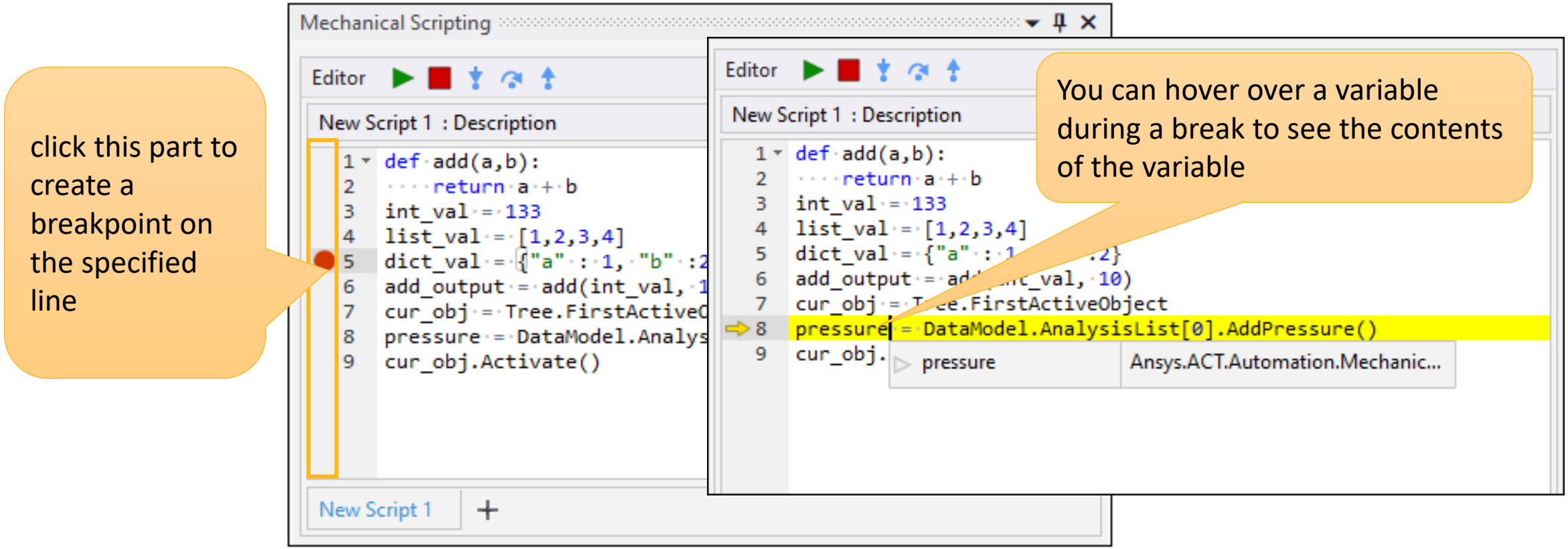
©2022 ANSYS, Inc.

Operation verification (how to use the debugger)

Ansys

Debugging features

- Assists in finding and correcting (debugging) errors in the created scripts.
- You can set breakpoints (where you want to pause) in the script or run it gradually (step in).



©2022 ANSYS, Inc.

If an error statement appears

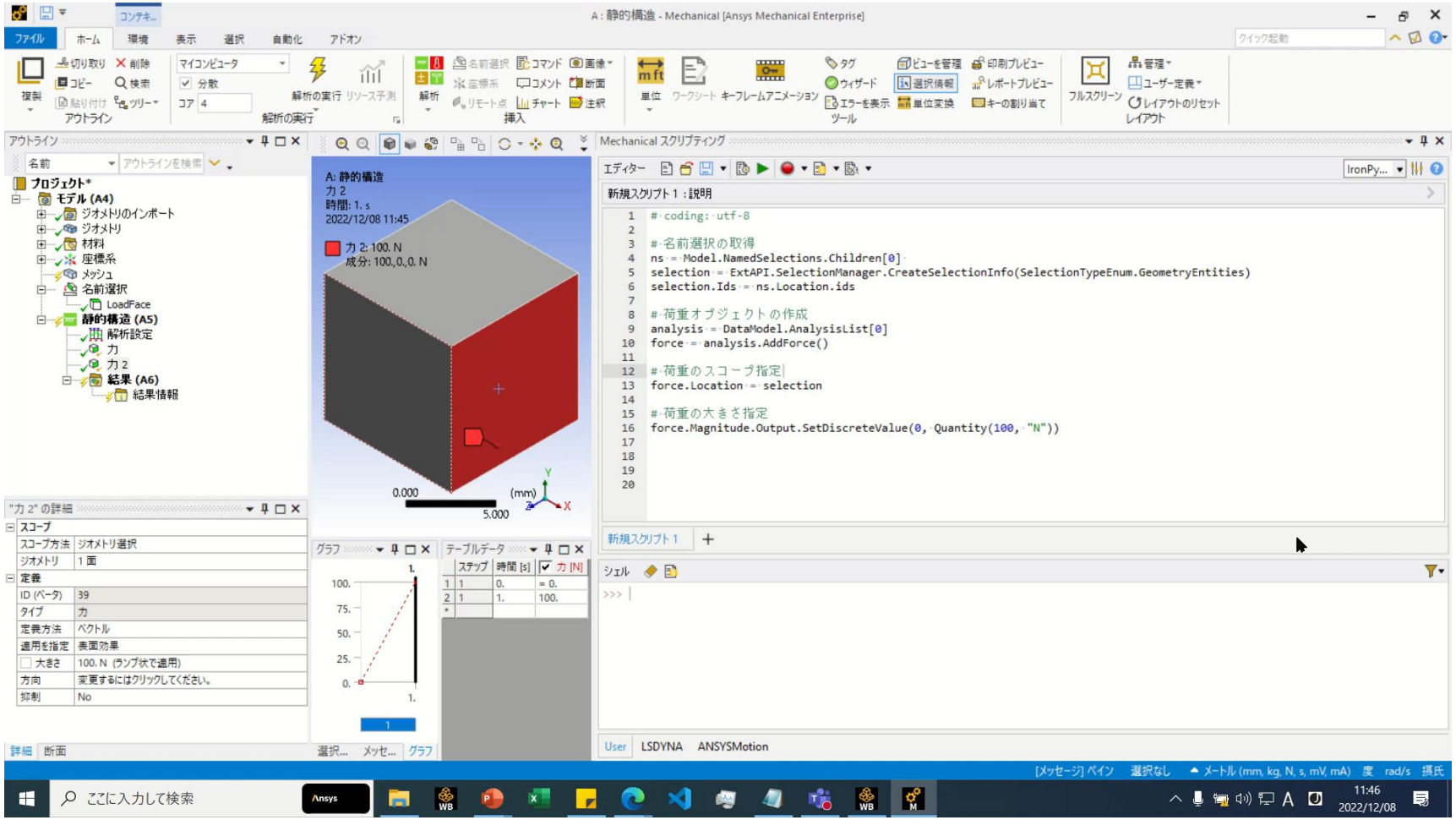
- If you get an error sentence in English, you're rather lucky! The scariest thing is the error-free bug.
- Don't be in a hurry just because it's an error and check the contents carefully.
- If you can understand the content properly, you can fix the problem smoothly.

- Common errors and their causes
  - AttibuteError : Referencing non-existent attributes (methods and properties), assigning Get-only properties
  - TypeError : Using variables of types that do not correspond to functions or operations
  - IndexError : You specified an index outside the range for the list, etc.
  - NameError: Forgot variable definition, forgot to import spelling mistakes

Future information on error-prone content and countermeasures will be published on the Ansys Customer Portal

©2022 ANSYS, Inc.

Specifically... Debugging and user buttons



©2022 ANSYS, Inc.

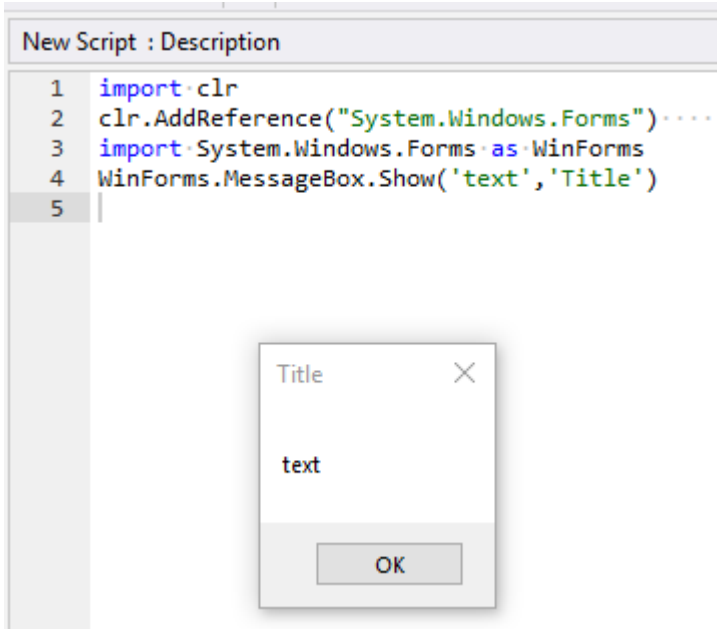
Advance tips tricks

Ansys

Advanced Tips and Tricks

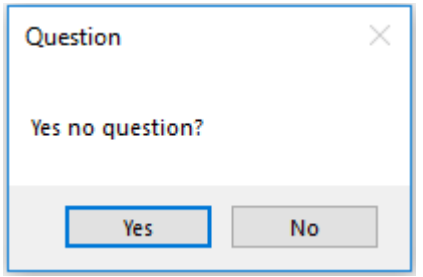
Message box
To display a message box you can use:

```
import clr
clr.AddReference("System.Windows.Forms")
import System.Windows.Forms as WinForms
WinForms.MessageBox.Show('text','Title')
```



from System.Windows.Forms import *

```
dialogResult = WinForms.MessageBox.Show("Yes no question?", "Question",
MessageBoxButtons.YesNo)
```

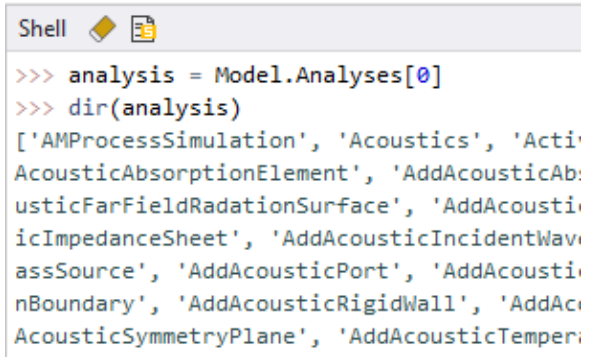


©2022 ANSYS, Inc.

Advanced Tips and Tricks

The dir command

- Sometimes you may want to list the items in the autocomplete. Do you so you can use the dir(...) command.
- For example everything which may come after analysis. Will be listed by dir(analysis)
- The dir command will typically list more items than the autocomplete.



©2022 ANSYS, Inc.

Advanced Tips and Tricks

Solve an APDL input file in batch mode

The console can be used to solve an APDL input file in batch mode

```
Import ansys
commandlinestring = " -b nolist -i "+“inpufile.inp"+" -o "+“outputfile.out"
ansys.RunANSYS(ExtAPI,commandlinestring, InputFileDirectory,minimized=True,hidden=True)
```

So for example to solve the current analysis in batch mode:

```
import ansys
analysis = Model.Analyses[0]
InputFileDirectory=analysis.WorkingDir # we are solving in the solver directory
analysis.WriteInputFile(InputFileDirectory+'inputfile.inp')# we generate the input file
commandlinestring = " -b nolist -i "+"inpufile.inp"+" -o "+"outputfile.out"
ansys.RunANSYS(ExtAPI,commandlinestring, InputFileDirectory,minimized=True,hidden=True)
```


©2022 ANSYS, Inc.

Advanced Tips and Tricks

Read and write files

To ask the user to browse a file:

File=ExtAPI.UserInterface.UIRenderer.ShowFileOpenDialog()

We can then read the selected file by using:

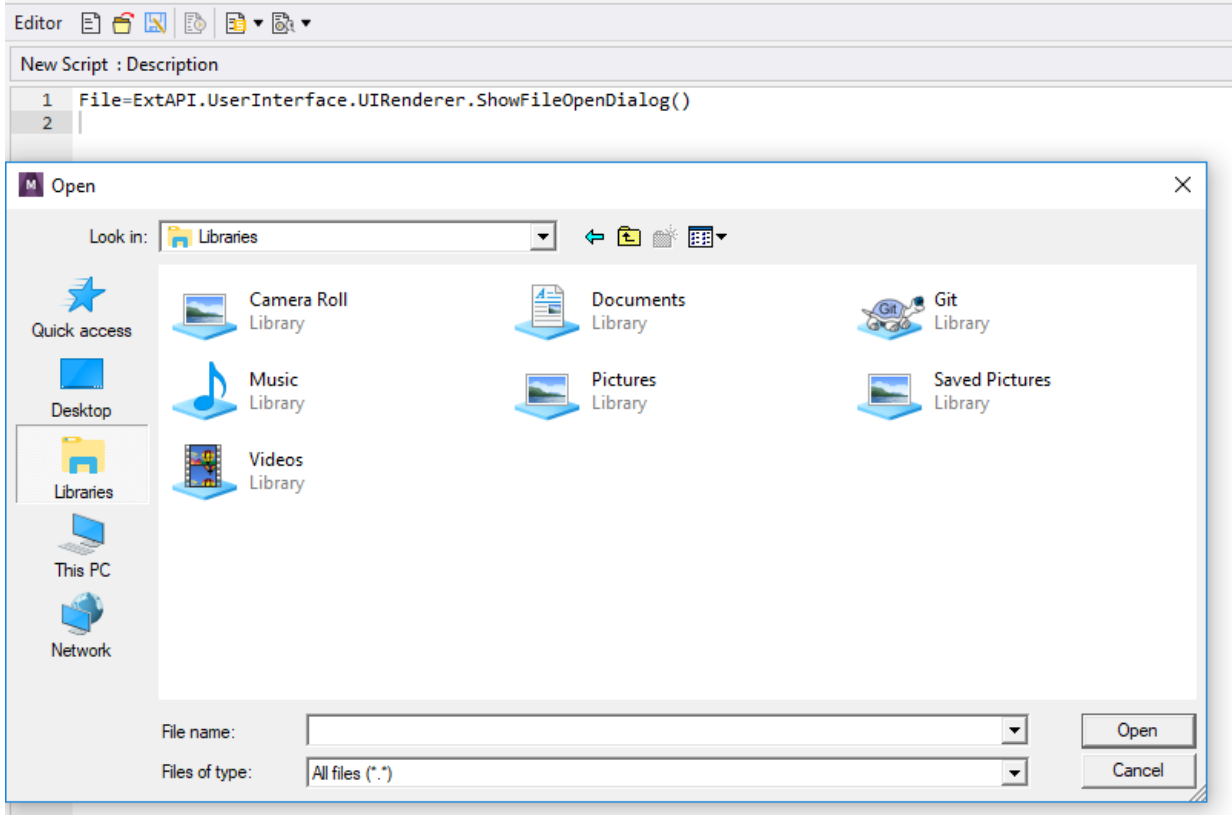
```
f=open(File,'r')
for line in f:
    print line
    ......

f.close()
```

In the same way we can write using in a file using:

```
f=open(File,'w')
f.write("some text....."  +"\n")
f.close()
```

"\n" denote the end of a line

Module 4 explains how to write directly to a *.csv file instead of a general file

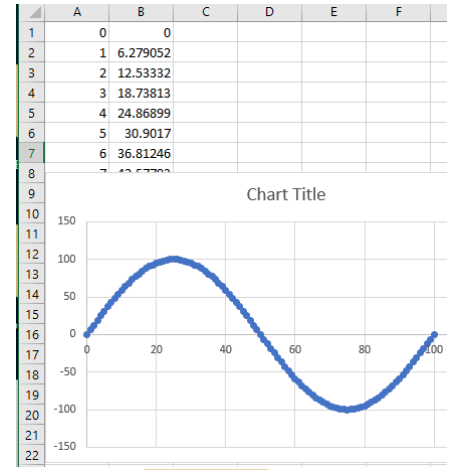


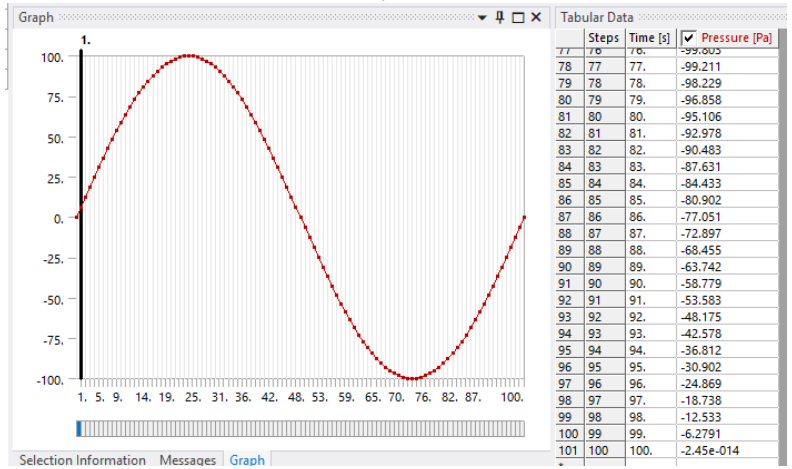
©2022 ANSYS, Inc.

Advanced Tips and Tricks

Read file example:

We assume that we have a csv file. The first column has the time data and the second column has the pressure data. We want the user to browse the csv and the pressure boundary condition will be applied for each time






```
File=ExtAPI.UserInterface.UIRenderer.ShowFileOpenDialog() #browse file
#read file and store values
t=list()
p=list()
f=open(File,'r')
for line in f:
    t.append(float(line.Split(',')[0]))
    p.append(float(line.Split(',')[1]))
f.close()

pressure = Model.Analyses[0].AddPressure() # Add a pressure
part1 = Model.Geometry.Children[0]  # Get the first part
body1 = part1.Children[0]  # Get the first body
face1 = body1.GetGeoBody().Faces[0]  # Get the first face of the body
selection = ExtAPI.SelectionManager.CreateSelectionInfo(SelectionTypeEnum.GeometryEntities)
selection.Entities = [face1]  #  Add the face to the selection
pressure.Location = selection  # Assign the selection to the pressure
pressure.Magnitude.Inputs[0].DiscreteValues = [Quantity(str(i) + "[s]") for i in t]  # Set the time values
pressure.Magnitude.Output.DiscreteValues = [Quantity(str(i)+ "[Pa]") for i in p]  # Set the magnitudes
analysis.AnalysisSettings.NumberOfSteps=len(t)-1#make the number of step the same as the number of points
```

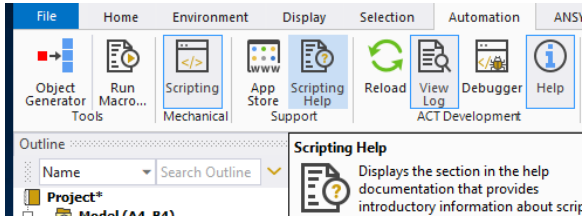

©2022 ANSYS, Inc.

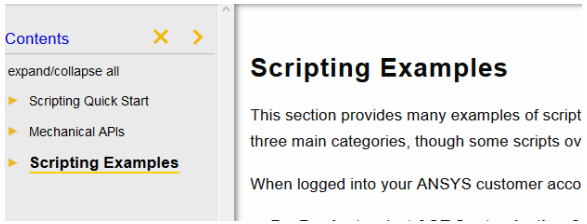
Advanced Tips and Tricks

The Help

- You can access the scripting help under Automation► Scripting help.

- The help contains a variety of examples which can be found under example or by using this link .





©2022 ANSYS, Inc.

PyAnsys

Ansys

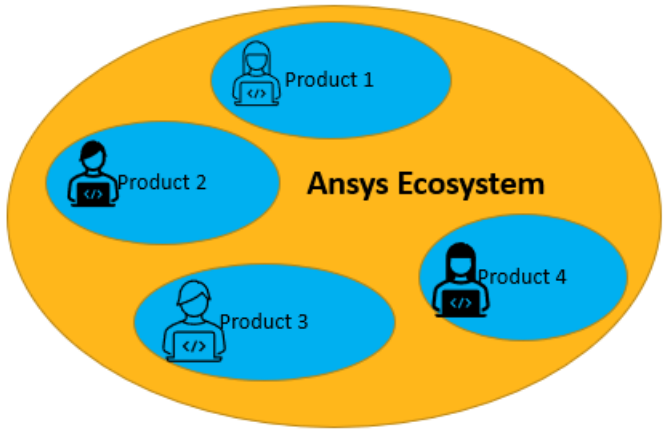
What is PyAnsys?

Python + **Ansys** = **PyAnsys**

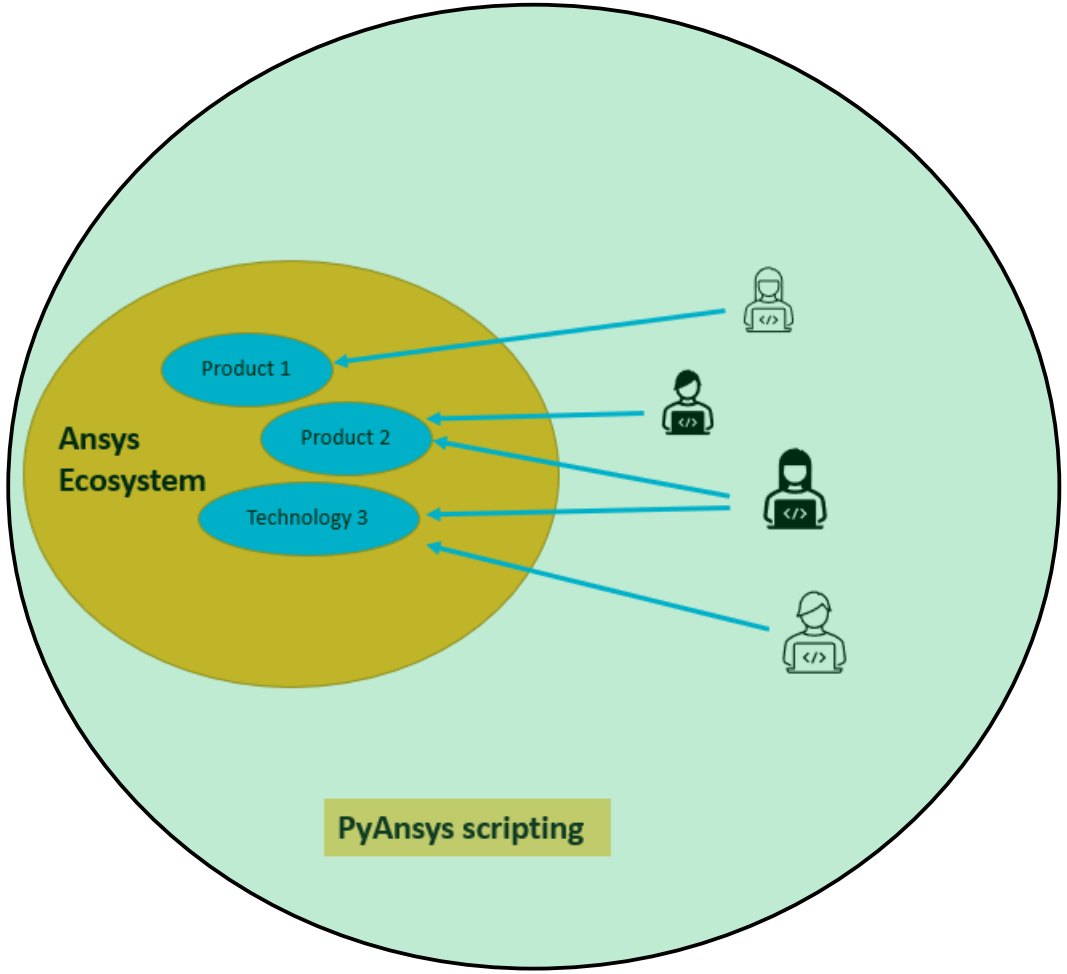
Set of technologies that allow the user to interface with Ansys products pythonically

©2022 ANSYS, Inc.

PyAnsys scripting



Product scripting

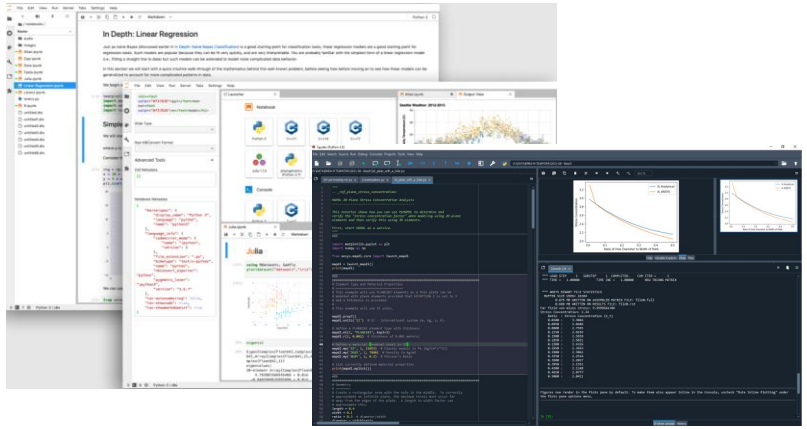


PyAnsys scripting

©2022 ANSYS, Inc.

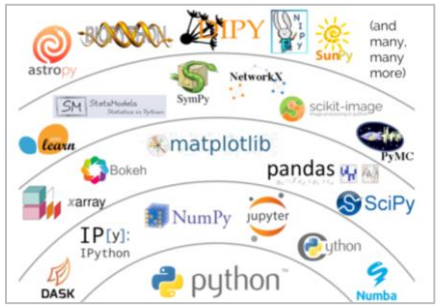
Scripting from within **YOUR** tools

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



Customer tools

Possibility to use other useful Python tools



PyAnsys technology

Ansys Technology

Products

- API — Ansys Product A
- API — Ansys Product B
- API — Ansys MAPDL

Technology components

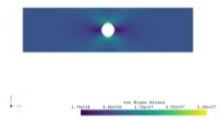
- Meshing
- Data Processing Framework

©2022 ANSYS, Inc.

What Packages Are Available?

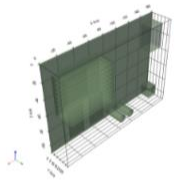
Check out all available packages in the PyAnsys doc

Simulation Libraries



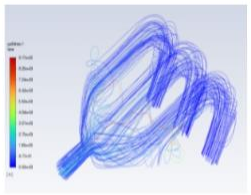
PyMAPDL

Pythonic interface to Ansys MAPDL (Mechanical APDL)



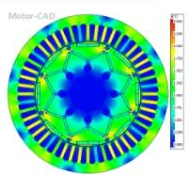
PyAEDT

Pythonic interface to AEDT (Ansys Electronic Desktop)



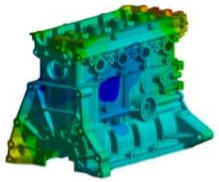
PyFluent

Pythonic interface to Ansys Fluent



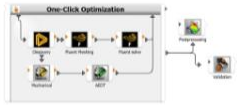
PyMotorCAD

Pythonic interface to Ansys Motor-CAD



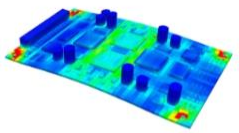
PyMechanical

Pythonic interface to Ansys Mechanical



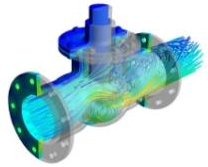
PyOptislang

Pythonic interface to Ansys Optislang



PySherlock

Pythonic interface to communicate with Ansys Sherlock



PySystem Coupling

Pythonic interface to communicate with Ansys System Coupling

PyDyna

Pythonic interface to build the Ansys DYNA input deck, submit it to the Ansys LS-DYNA solver and postprocess its results

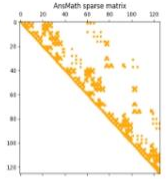
Utility Libraries

PyPrimeMesh

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

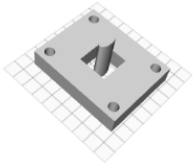
PyDynamicReporting

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



PyAnsys Math

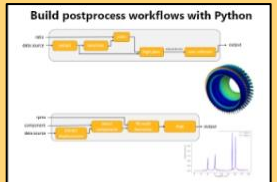
Pythonic interface to PyAnsys Math libraries



PyAnsys Geometry

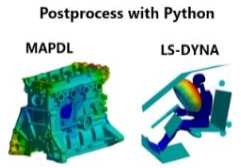
Pythonic interface to the Ansys Geometry service

Postprocessing Libraries



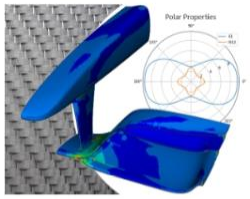
PyDPF - Core

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



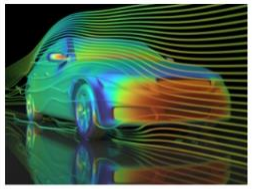
PyDPF - Post

Pythonic interface to access and post process Ansys solver result files



PyDPF - Composites

Pythonic interface to post-process layered and short-fiber composite models



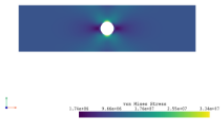
PyEnSight

Pythonic interface to EnSight, the Ansys simulation postprocessor

©2022 ANSYS, Inc.

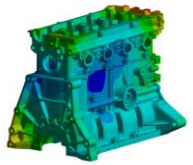
PyAnsys: What's available?

Simulation libraries



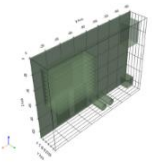
PyMAPDL

Pythonic interface to Ansys MAPDL (Mechanical APLD)



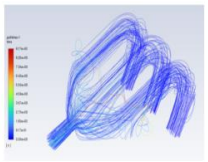
PyMechanical

Pythonic interface to Ansys Mechanical



PyAEDT

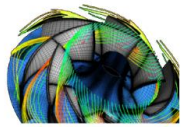
Pythonic interface to AEDT (Ansys Electronic Desktop)



PyFluent

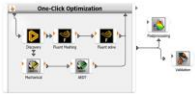
Pythonic interface to Ansys Fluent

Utility libraries



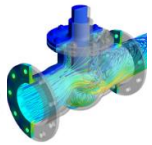
PyPrimeMesh

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



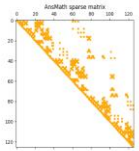
PyOptislang

Pythonic interface to Ansys Optislang



PySystem Coupling

Pythonic interface to communicate with Ansys System Coupling



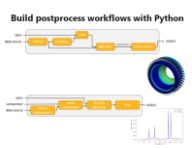
PyAnsys Math

Pythonic interface to PyAnsys Math libraries

PyDynamicReporting

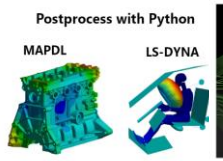
Pythonic interface to Ansys Dynamic Reporting for service and control of its database and reports

Post-processing libraries



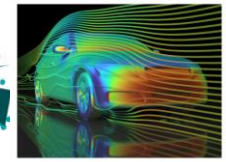
PyDPF - Core

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



PyDPF - Post

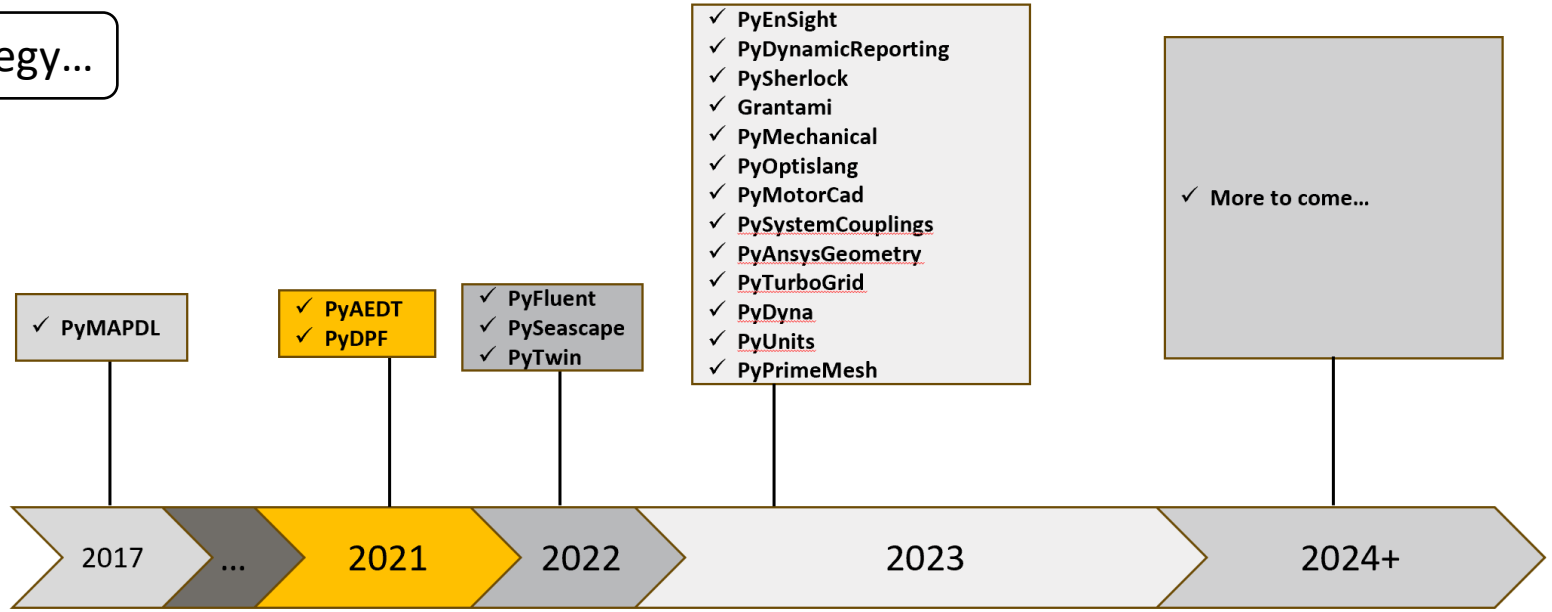
Pythonic interface to access and post process Ansys solver result files



PyEnSight

Pythonic interface to EnSight, the Ansys simulation postprocessor

5-year strategy…

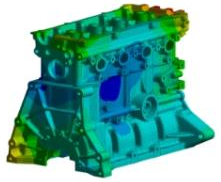


Keep an eye on…
docs.pyansys.com

©2022 ANSYS, Inc.

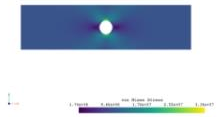
And If I'm a Structural Engineer?

Simulation Libraries



PyMechanical

Pythonic interface to Ansys Mechanical



PyMAPDL

Pythonic interface to Ansys MAPDL (Mechanical APDL)

PyDyna

Pythonic interface to build the Ansys DYNA input deck, submit it to the Ansys LS-DYNA solver and postprocess its results

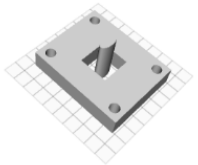
Utility Libraries

PyPrimeMesh

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

PyDynamicReporting

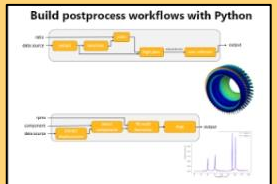
Pythonic interface to Ansys Dynamic Reporting for service and control of its database and reports



PyAnsys Geometry

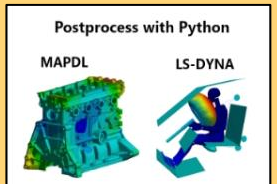
Pythonic interface to the Ansys Geometry service

Postprocessing Libraries



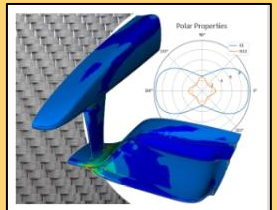
PyDPF - Core

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

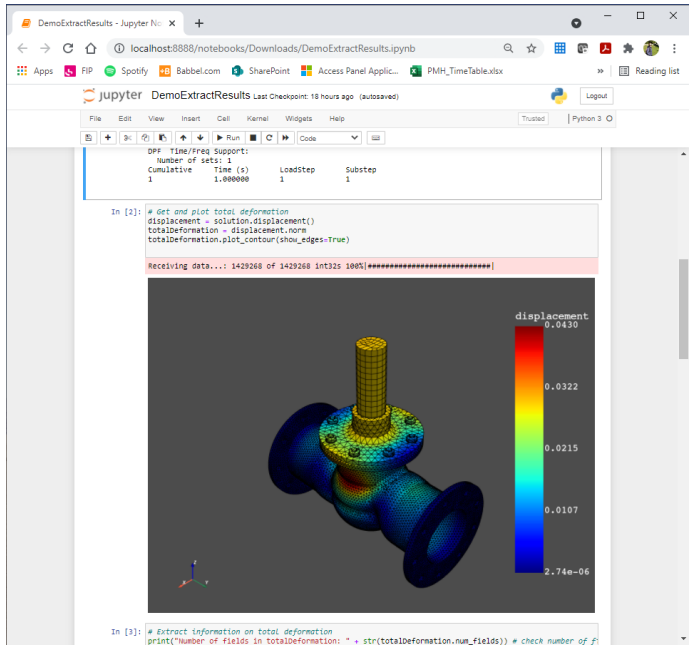


PyDPF - Composites

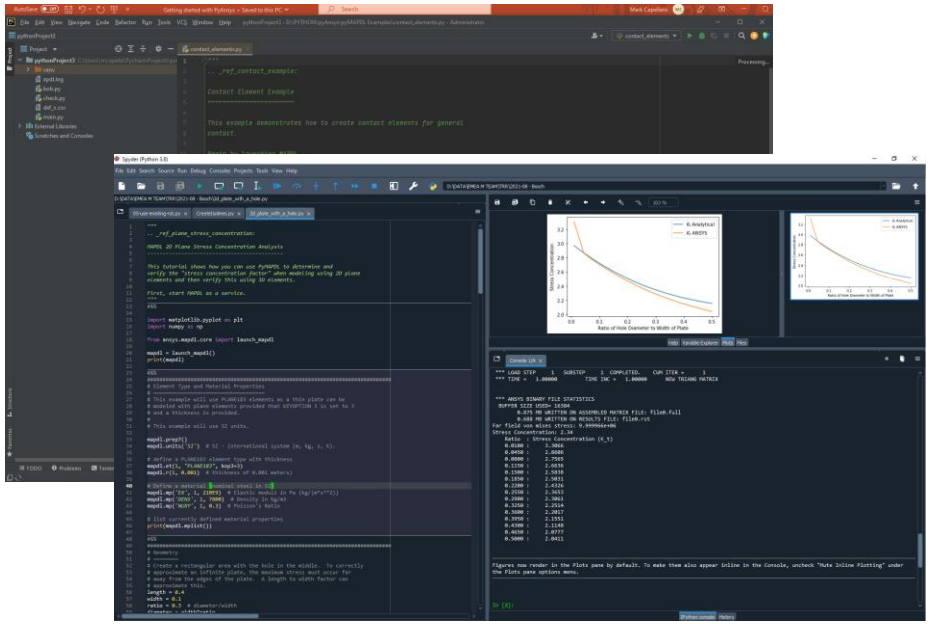
Pythonic interface to post-process layered and short-fiber composite models

©2022 ANSYS, Inc.

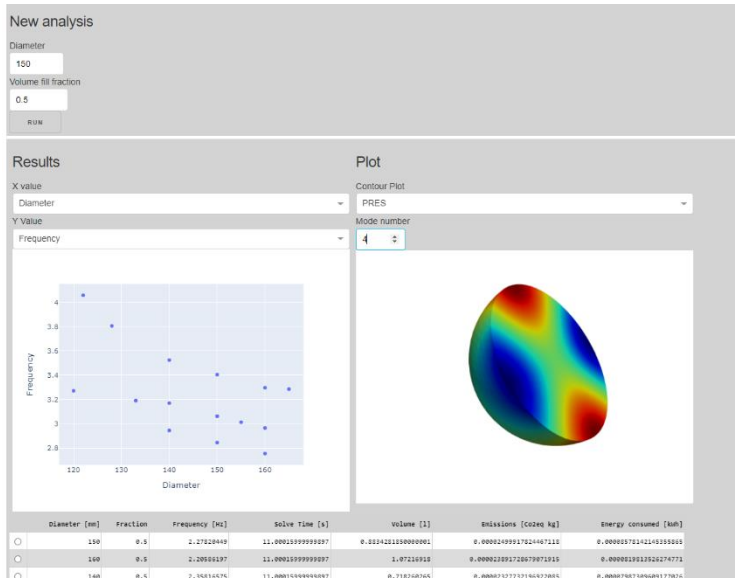
Is there a user interface (UI)?



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

©2022 ANSYS, Inc.

PyMechanical

- Control Mechanical from Python

Ansys

Python scripting for Mechanical

1. Classic (IronPython)
   - Run inside the scripting window (ACT API IronPython)
2. Embedded instance (CPython)
   - An instance of mechanical is embedded into the python process as an object using PyMechanical
3. Remote session (IronPython server, CPython client)
   - Start a remote session of Mechanical in its own process
     - The remote session can be running on the local machine.
   - Connect to that session from CPython using PyMechanical and send commands as IronPython strings or scripts

©2022 ANSYS, Inc.

PyMechanical: access Mechanical through Python

install via:

```
pip install ansys-mechanical-core
```

Remote session

- Multiple programs:
  - Server = Mechanical
  - Client = Python (PyMechanical)
- Client can send commands to the server as strings, or as .py scripts
- Based on gRPC

Embedded instance

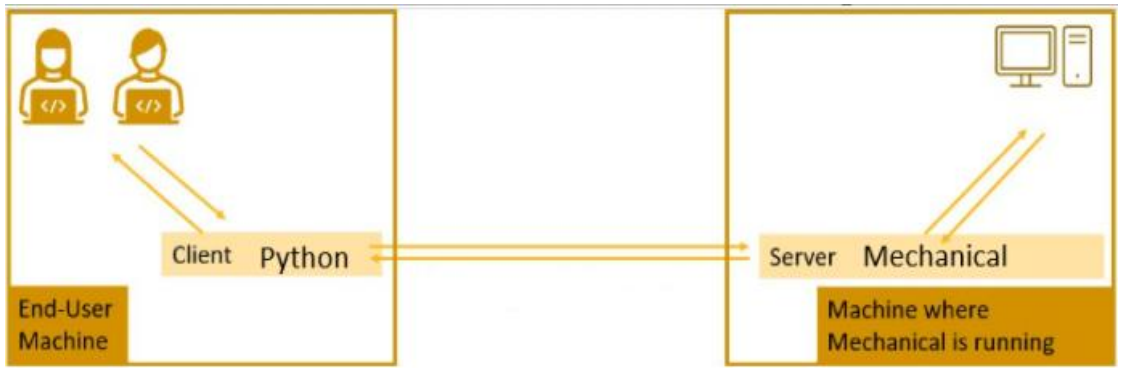
- Python process only
- Mechanical application instance as a python object
- Use in the same way as Mechanical Scripting
- Based on pythonnet

Go here for an explanation of the Mechanical application architecture and why both interfaces are offered

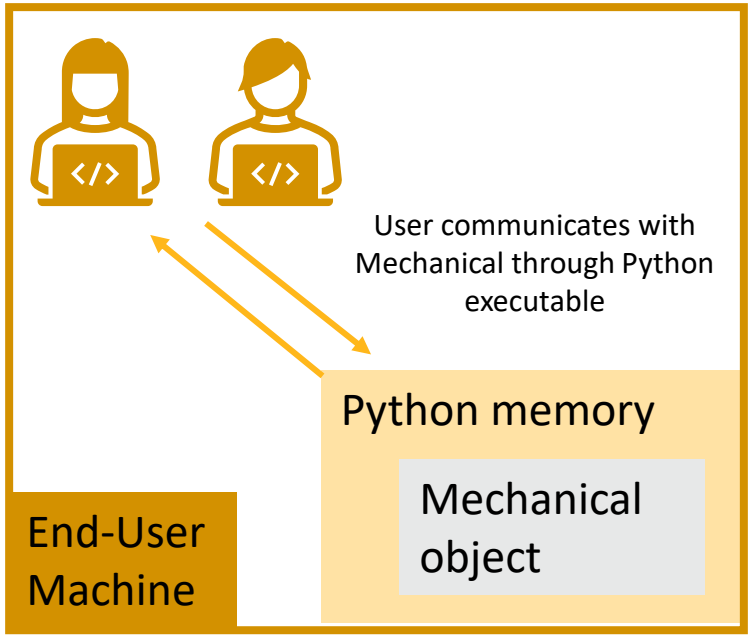
©2022 ANSYS, Inc.

PyMechanical: access Mechanical through Python

Remote session



Embedded instance



©2022 ANSYS, Inc.

Remote session

Launch a session locally:

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

mechanical = launch_mechanical()
```

```python
preprocess = mechanical.run_python_script(
    """


# Section 1: Read geometry information
geometry_import_group = Model.GeometryImportGroup
geometry_import = geometry_import_group.AddGeometryImport()
geometry_import_format = (
    Ansys.Mechanical.DataModel.Enums.GeometryImportPreference.Format.Automatic
)
geometry_import_preferences = Ansys.ACT.Mechanical.Utilities.GeometryImportPreferences()
geometry_import_preferences.ProcessNamedSelections = True
geometry_import.Import(
    geometry_path, geometry_import_format, geometry_import_preferences
)
# Import materials.
MAT = ExtAPI.DataModel.Project.Model.Materials
MAT.Import(mat_file_path)

Model.AddStaticStructuralAnalysis()
ExtAPI.Application.ActiveUnitSystem = MechanicalUnitSystem.StandardNMM

#Material assignment
for Geo in Model.Geometry.GetChildren(DataModelObjectCategory.Body, True):
    #Geo=Assemble.Children[0]
    if 'valve' in Geo.Name or 'flange' in Geo.Name:
        Geo.Material='Type 40 Gray Cast Iron'
    elif 'seal' in Geo.Name:
        Geo.Material='Type 302 Stainless Steel'
    else:
        Geo.Material='AISI 6150 Steel'
```

Main methods:

`Mechanical.clear`()	Clear the database.
`Mechanical.download`(files[, target_dir, ...])	Download files from the working directory of the Mechanical instance.
`Mechanical.download_project`([extensions, ...])	Download all project files in the working directory of the Mechanical instance.
`Mechanical.exit`([force])	Exit Mechanical.
`Mechanical.list_files`()	List the files in the working directory of Mechanical.
`Mechanical.log_message`(log_level, message)	Log the message using the given log level.
`Mechanical.project_directory`	Get the project directory for the currently connected Mechanical instance.
`Mechanical.run_python_script`(script_block[, ...])	Run a Python script block inside Mechanical.
`Mechanical.run_python_script_from_file`(file_path)	Run a Python file inside Mechanical.
`Mechanical.upload`(file_name[, ...])	Upload a file to the Mechanical instance.
`Mechanical.version`	Get the Mechanical version based on the instance.

©2022 ANSYS, Inc.

Embedded instance

Launch an embedded instance:

```
from ansys.mechanical.core import App

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

Entry points (same as in-product scripting):

ExtAPI: `Application.ExtAPI`

DataModel: `Application.DataModel`

Model: `Application.DataModel.Project.Model`

Tree: `Application.DataModel.Tree`

Graphics: `Application.ExtAPI.Graphics`

Useful tip: import additional entry points, namespaces, and types (Quantity, DataModelObjectCategory, etc.):

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

app = App()
# The following line extracts the global API entry points and merges them into your global
# Python global variables.
globals().update(global_variables(app))
```

```
MAT = Model.Materials
MAT.Import(mat_path)

# sphinx_gallery_start_ignore
#assert str(MAT.ObjectState) == "FullyDefined", "Materials are not defined"
# sphinx_gallery_end_ignore

Model.AddStaticStructuralAnalysis()

ExtAPI.Application.ActiveUnitSystem = MechanicalUnitSystem.StandardNMM
#Preprocess are same with Mechanical ACTAPI
#Material assignment
for Geo in Model.Geometry.GetChildren(DataModelObjectCategory.Body, True):
    #Geo=Assemble.Children[0]
    if 'valve' in Geo.Name or 'flange' in Geo.Name:
        Geo.Material='Type 40 Gray Cast Iron'
    elif 'seal' in Geo.Name:
        Geo.Material='Type 302 Stainless Steel'
    else:
        Geo.Material='AISI 6150 Steel'
#Create Nameselection by worksheet
for i in range(2):
    NS=Model.AddNamedSelection()
    NS.ScopingMethod=GeometryDefineByType.Worksheet
    NS.Name="Fixed"+str(i)
    NSWS=NS.GenerationCriteria
    NSWS.Add(None)
    NSWS[0].EntityType=SelectionType.GeoFace
    NSWS[0].Criterion=SelectionCriterionType.LocationY
    if i==0:
        NSWS[0].Operator=SelectionOperatorType.Largest
    else:
```

©2022 ANSYS, Inc.

Learning flow and How to use the document

Ansys

How to use the learning flow and documentation

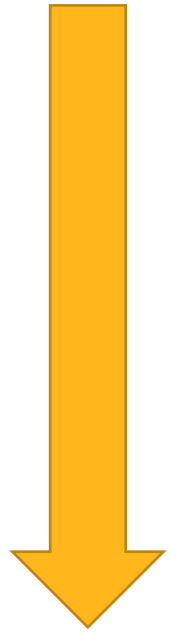
If you imagine learning a language...

- Concept Understanding (Textbook)
  - Mechanical Scripting Guide(ansys.com)com)
  - Structure | Customization - Getting Started with Ansys ACT Mechanical(sapjam.com) (Separate contract required)
  - ACT Developer's Guide (ansys.com)

- Actually try it (reference book / problem collection)
  - Part III Example script(ansys.com)
  - Search by Script/Script in Ansys Customer Portal

- Modify the code and customize it or rewrite it to another function (dictionary)
  - Ansys ACT 2022 R2 Release API and XML Online Reference Guide for Mechanical

conspectus



detail

©2023 ANSYS, Inc. / Confidential

Python基本語法簡介(1/4)

https://www.tutorialspoint.com/python/index.htm

- Python - Basic Syntax
- Python - Variable Types

©2023 ANSYS, Inc. / Confidential

Python Keyword

- Keyword(保留字)不能用於變數或函數命名。

種類	關鍵字
常數	`False` `None` `True`
運算子	`and` `del` `in` `is` `lambda` `not` `or`
簡單陳述	`as` `assert` `break` `continue` `from` `global` `import` `nonlocal` `pass` `raise` `return` `yield`
複合陳述	`else` `elif` `except` `finally` `for` `if` `try` `while` `with`
定義	`class` `def`

©2023 ANSYS, Inc. / Confidential

縮排

Python透過縮排來組織程式碼並提高可讀性。在控制結構的複合陳述會需要用到：

- **If …: else:**
- **for … in …:**
- **def foo():**
- **class foo():**
- **with open():**

縮排規則如下：

- 「:」之後的下一行必須縮排，除非程式碼緊接在「:」之後的同一行且只有一行
- 慣例縮排的空白數為4，同一個縮排等級的程式碼必須使用相同的空白數進行縮排
- 違反縮排規則會出現**IndentationError**

©2023 ANSYS, Inc. / Confidential

註解

- 在電腦語言中，註解是電腦語言的一個重要組成部分，用於在原始碼中解釋代碼的功用，可以增強程式的可讀性，可維護性。
- 如果沒辦法使用少量的單字為變數命名，或是用適當的動詞作為函式的命名，就得用註解完整描述程式碼。換句話說，如果可以透過命名解釋意圖，則不用註解解釋程式碼的意圖。
- 請使用英文註解。中文註解在AEDT執行時會產生編碼錯誤使得程式無法執行。

- 單行

```
#This is comment
```

- 多行：

```
''' comment1
comment2
comment3'''
```


©2023 ANSYS, Inc. / Confidential

變數型別與宣告

Python基本資料型別

- Numbers: x = 3, x = 1.34, x = 3+4j
- String: x = “Hello World”, x = ‘Hello World’, x = ‘”Hello”, he said’, x=‘3.14’
- List: x = [1,2,3], x = [‘A’, ‘B’, ‘C’]
- Tuple: x = (‘John’, 37, “Male”)
- Dictionary: x = {‘John’: (37, ‘Male’), ‘Mary’:(18, ‘Female’) }
- Boolean: True, False
  - 判斷為False
    - None
    - 數字0
    - 空字串及空資料，如：‘’、[]、()、 {}
  - 其他判斷為True

©2023 ANSYS, Inc. / Confidential

Python運算子類型

可分成下面幾類:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators(自動化不常用)
- Membership operators
- Bitwise operators(自動化不常用)

©2023 ANSYS, Inc. / Confidential

算術運算子(Arithmetic operators)

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

©2023 ANSYS, Inc. / Confidential

指派運算子(Assignment operators)

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

©2023 ANSYS, Inc. / Confidential

比較運算子(Comparison operators)

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

©2023 ANSYS, Inc. / Confidential

成員運算子(Membership operators)

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

©2023 ANSYS, Inc. / Confidential

邏輯運算子(Logical Operator)

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

©2023 ANSYS, Inc. / Confidential

Python基本語法簡介(2/4)

https://www.tutorialspoint.com/python/index.htm

- Python - Basic Operators
- Python - Decision Making
- Python - Loops
- Python - Numbers
- Python - Strings

©2023 ANSYS, Inc. / Confidential

常用内建函式

- abs()
- complex()
- dict()
- dir()
- enumerate()
- filter()
- float()
- int()
- len()
- list()
- map()
- max()
- min()
- open()
- pow()
- print()
- range()
- round()
- str()
- sum()
- tuple()
- type()
- zip()

Built-in Functions				
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

©2023 ANSYS, Inc. / Confidential

f = open() 與 with open() as f:

```
>>>
>>> f = open('d:/demo2/locations.csv')
>>> text = f.readlines()
>>> text
['1, 0 ,0\n', '0, 1, 0\n', '0,0,1\n', '2,0,0\n', '3,0,0\n']
>>> f.close()
>>> with open('d:/demo2/locations.csv') as f:
...     text2 = f.readlines()
...
>>> text2
['1, 0 ,0\n', '0, 1, 0\n', '0,0,1\n', '2,0,0\n', '3,0,0\n']
>>>
```


©2023 ANSYS, Inc. / Confidential

map(), filter() 與 list comprehension

```
>>> x = ['-2', '3', '5', '-6', '1', '-4', '0']
>>> y = map(float, x)
>>> y
[-2.0, 3.0, 5.0, -6.0, 1.0, -4.0, 0.0]
>>> def greater_than_zero(num):
...     return num > 0
...
>>> z = filter(greater_than_zero, y)
>>> z
[3.0, 5.0, 1.0]
>>> w = [float(i) for i in x if float(i)>0]
>>> w
[3.0, 5.0, 1.0]
>>>
```


©2023 ANSYS, Inc. / Confidential

IronPython啟動開檔瀏覽對話框

```
import sys
import clr
clr.AddReference("System.Windows.Forms")

from System.Windows.Forms import DialogResult, OpenFileDialog
dialog = OpenFileDialog()
dialog.Filter = "text files (*.txt)|*.txt"

if dialog.ShowDialog() == DialogResult.OK:
    txt_path = dialog.FileName
    AddWarningMessage(txt_path)
else:
    pass
```


©2023 ANSYS, Inc. / Confidential

字串的索引與切片

切片是相當方便的字串處理工具之一，利用中括號來取出部分字串或倒置

- x[start:stop:step]

```
In [8]: x='channel.s4p'

In [9]: x[0]
Out[9]: 'c'

In [10]: x[1:7]
Out[10]: 'hannel'

In [11]: x[-3:]
Out[11]: 's4p'

In [12]: x[:-4]
Out[12]: 'channel'

In [13]: x[::-1]
Out[13]: 'p4s.lennahc'

In [14]: x[::2]
Out[14]: 'canlsp'
```


©2023 ANSYS, Inc. / Confidential

Escape Characters

```
>>> print('Hello World')
Hello World
>>> print('Hello\nWorld')
Hello
World
>>> print('Hello\\World')
Hello\World
>>> print('1\t2\t3')
1	2	3
>>>
```

Code	Result	Try it
\'	Single Quote	Try it »
\\	Backslash	Try it »
\n	New Line	Try it »
\r	Carriage Return	Try it »
\t	Tab	Try it »
\b	Backspace	Try it »
\f	Form Feed	
\ooo	Octal value	Try it »
\xhh	Hex value	Try it »

如果在字串中要維持”\n” 可以在字串符前加上r

©2023 ANSYS, Inc. / Confidential

常用字串運算

```
>>> x='Channel.s36p'
>>> x.split('.')
['Channel', 's36p']
>>> 'd:/demo/'+x
'd:/demo/Channel.s36p'
>>> x.replace('.s', '_new.s')
'Channel_new.s36p'
>>> x.upper()
'CHANNEL.S36P'
>>> x.lower()
'channel.s36p'
>>> '.s' in x
True
>>> len(x)
12
>>> dir(x)
['Chars', 'Clone', 'Compare', 'CompareOrdinal', 'CompareTo', 'Concat', 'Contains', 'Copy', 'CopyTo',
'Empty', 'EndsWith', 'Equals', 'Format', 'GetEnumerator', 'GetHashCode', 'GetType', 'GetTypeCode',
'IndexOf', 'IndexOfAny', 'Insert', 'Intern', 'IsInterned', 'IsNormalized', 'IsNullOrEmpty',
'IsNullOrWhiteSpace', 'Join', 'LastIndexOf', 'LastIndexOfAny', 'Length', 'MemberwiseClone', 'Normalize',
'PadLeft', 'PadRight', 'ReferenceEquals', 'Remove', 'Replace', 'Split', 'StartsWith', 'Substring',
'ToCharArray', 'ToLower', 'ToLowerInvariant', 'ToString', 'ToUpper', 'ToUpperInvariant', 'Trim',
'TrimEnd', 'TrimStart', '__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__',
'__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__',
'__radd__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split', '_formatter_parser',
'capitalize', 'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format',
'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'islower', 'isnumeric', 'isspace', 'istitle',
'isunicode', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex',
'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase',
'title', 'translate', 'upper', 'zfill']
```


©2023 ANSYS, Inc. / Confidential

分解與組合字串

```
>>> x = ['box1', 'cylinder1', 'sphere1']
>>> y = ','.join(x)
>>> print(y)
box1,cylinder1,sphere1
>>> type(y)
<type 'str'>
>>> z = y.split(',')
>>> z
['box1', 'cylinder1', 'sphere1']
>>>
```


©2023 ANSYS, Inc. / Confidential

邏輯判斷If, else, elif

Python 提供了 if 、 else 、 elif 這三種語法來協助我們實現各種條件判斷和流程控制。Python 程式語言是一行一行執行的，所以當我們想要所寫的程式在某些條件下跳過某幾行敘述、不再照單全收的時候，就可以使用條件判斷。也就是說，如果要讓程式可以自動檢查所處理資料的內容，而且根據資料內容來決定是否執行某一個敘述或指令，那就需要用到條件判斷式來控制流程。

求解公倍數

```
for i in range(1000):
    if i%21 == 0 and i%24 == 0:
        print(i)
```


©2023 ANSYS, Inc. / Confidential

三元運算子(ternary operator)

- 三元運算子將變量指定及判斷整合在一行當中
- 好處是程式碼比較簡短。

- 一般

```
if x > y:
    z = x
else:
    z = y
```

- 三元運算子

```
z = x if x > y else y
```


©2023 ANSYS, Inc. / Confidential

迴圈控制 for...in...:

- 單純的for迴圈會依序對列表當中每個元素做運算，直到所有元素都完成處理。加上break直接脫離迴圈，而continue敘述則是中斷這個元素後續工作進到下一個元素。這兩個敘述通常會搭配if-else使用。這裡用兩個例子來說明break以及continue的使用時機。

```
In [40]: for i in range(1, 1000):
    ...:     if i%21==0 and i%24==0:
    ...:         print(i)
    ...:         break
    ...:
168
```

```
In [42]: for i in range(1, 1000):
    ...:     if i%21==0 and i%24==0:
    ...:         continue
    ...:     else:
    ...:         print(i)
```


©2023 ANSYS, Inc. / Confidential

拆包

```
>>> x, y, z = (1, 2, 3)
>>> x
1
>>> y
2
>>> z
3
>>> shopping_list = [('Apple', 5.4, 6), ('Banana', 3.8, 12), ('lemon', 1.2, 30)]
>>> for name, price, number in shopping_list:
...     print(name+': $'+str(price*number))
...
Apple: $32.4
Banana: $45.6
lemon: $36.0
>>>
```


©2023 ANSYS, Inc. / Confidential

函數range()與enumerate()

這兩個函數經常與for合併使用

- 函數range()產生等差的整數數列。
- 當需要index來操作list時可使用range()來建立index。

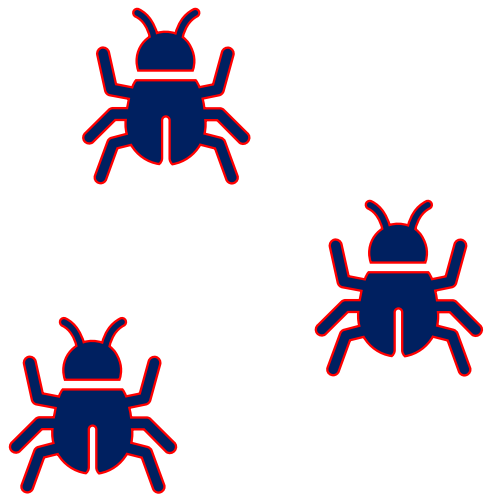
```
In [44]: list(range(10))
Out[44]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [45]: list(range(2, 10))
Out[45]: [2, 3, 4, 5, 6, 7, 8, 9]

In [46]: list(range(2, 10, 2))
Out[46]: [2, 4, 6, 8]
```

- enumerate多用於在for循環中得到計數，利用它可以同時獲得索引和值，即需要index和value值的時候可以使用enumerate。

```
In [50]: names = ['John', 'Mary', 'Anna', 'Ken']

In [51]: for n, i in enumerate(names):
    ...:     print(str(n) + ':' + i)
    ...:
0:John
1:Mary
2:Anna
3:Ken
```


©2023 ANSYS, Inc. / Confidential

除錯 (Debug)



寫程式是不斷的嘗試及除錯，程式錯誤基本上可分成三種：

- Syntax Error(語法錯誤)
- Run Time Error(執行時期錯誤)
- Logic Error(邏輯錯誤)

一般的IDE環境在編寫階段即可偵測Syntax Error 。Run Time Error通常是輸入資料的瑕疵所造成，通常是在執行過程會被檢查出來。可以透過例外處理來解決。至於Logic Error則是輸出結果不如預期，這個錯誤要修正則是困難許多。一般需要設定斷點觀察變量的變動來追蹤可能的邏輯錯誤。在AEDT當中只能透過輸出變數值或log檔來協助除錯。

©2023 ANSYS, Inc. / Confidential

Syntax Error (語法錯誤)

- 圖一的程式碼是要計算x當中所有數字的總和。在第10行for迴圈的最後少了冒號，這違反了Python的語法規則，因此編輯器在行數10之前顯示了叉叉符號，提醒開發者語法錯誤。
- 如果不修正直接執行，則會出現如圖二的語法錯誤訊息
- 在第十行for迴圈之後補上了冒號之後(如圖三)，叉叉符號消失，代表沒有語法錯誤。

```
 8 x = [1, 2, 3, 'John', 5, 6, 7]
 9 total = 0
10 for i in x
11     total += i
12
13 print(total)
14
```

```
  File "C:/Users/mlin/AppData/Roaming/SPB_Data/untitled16.py", line 10
    for i in x
              ^
SyntaxError: invalid syntax
```

```
 8 x = [1, 2, 3, 'John', 5, 6, 7]
 9 total = 0
10 for i in x:
11     total += i
12
13 print(total)
14
```


©2023 ANSYS, Inc. / Confidential

Run Time Error(執行時期錯誤)

- 此時再次執行程式，另一個錯誤出現(圖左)
- 這個錯誤是因為x當中混入了字串'John'，而字串是無法與數字相加的。由於這是無法事先預期的錯誤且在執行時期才發生，所以又稱作例外。這種錯誤可以用例外處理機制來解決，程式碼修改如圖右：

```
 8 x = [1, 2, 3, 'John', 5, 6, 7]
 9 total = 0
10 for i in x:
11     total += i
12
13 print(total)
14
```

```
 8 x = [1, 2, 3, 'John', 5, 6, 7]
 9 total = 0
10 for i in x:
11     try:
12         total += i
13     except:
14         pass
15 print(total)
16
```

```
TypeError: unsupported operand type(s) for +=: 'int' and 'str'
```

```
In [12]: runfile('C:/Users/mlin/AppData/Roaming/SPB_Data/untitled16.py',
wdir='C:/Users/mlin/AppData/Roaming/SPB_Data')
24
```


©2023 ANSYS, Inc. / Confidential

Logic Error (邏輯錯誤)

- 邏輯錯誤指的是代碼可以完成執行而不會返回錯誤訊息，然而執行結果卻不符合預期。左圖的程式碼目的是求取奇數項的和，即1+3+5+7，總和應為16。然而計算結果卻是12。
- 原因是Python的引數是從0開始，所以程式碼必須改成sum(x[0::2])，才能得到預期的結果16。邏輯錯誤一般來說要遠比語法錯誤及執行時期錯誤錯誤難處理的多。

```
 8 x = [1, 2, 3, 4, 5, 6, 7]
 9 total = 0
10 print(sum(x[1::2]))
11
```

```
In [15]: runfile('C:/Users/mlin/AppData/Roaming/SPB_Data/untitled16.py',
wdir='C:/Users/mlin/AppData/Roaming/SPB_Data')
12
```

```
 8 x = [1, 2, 3, 4, 5, 6, 7]
 9 total = 0
10 print(sum(x[0::2]))
11
```

```
In [16]: runfile('C:/Users/mlin/AppData/Roaming/SPB_Data/untitled16.py',
wdir='C:/Users/mlin/AppData/Roaming/SPB_Data')
16
```


©2023 ANSYS, Inc. / Confidential

常見Exception

IronPython Command Window

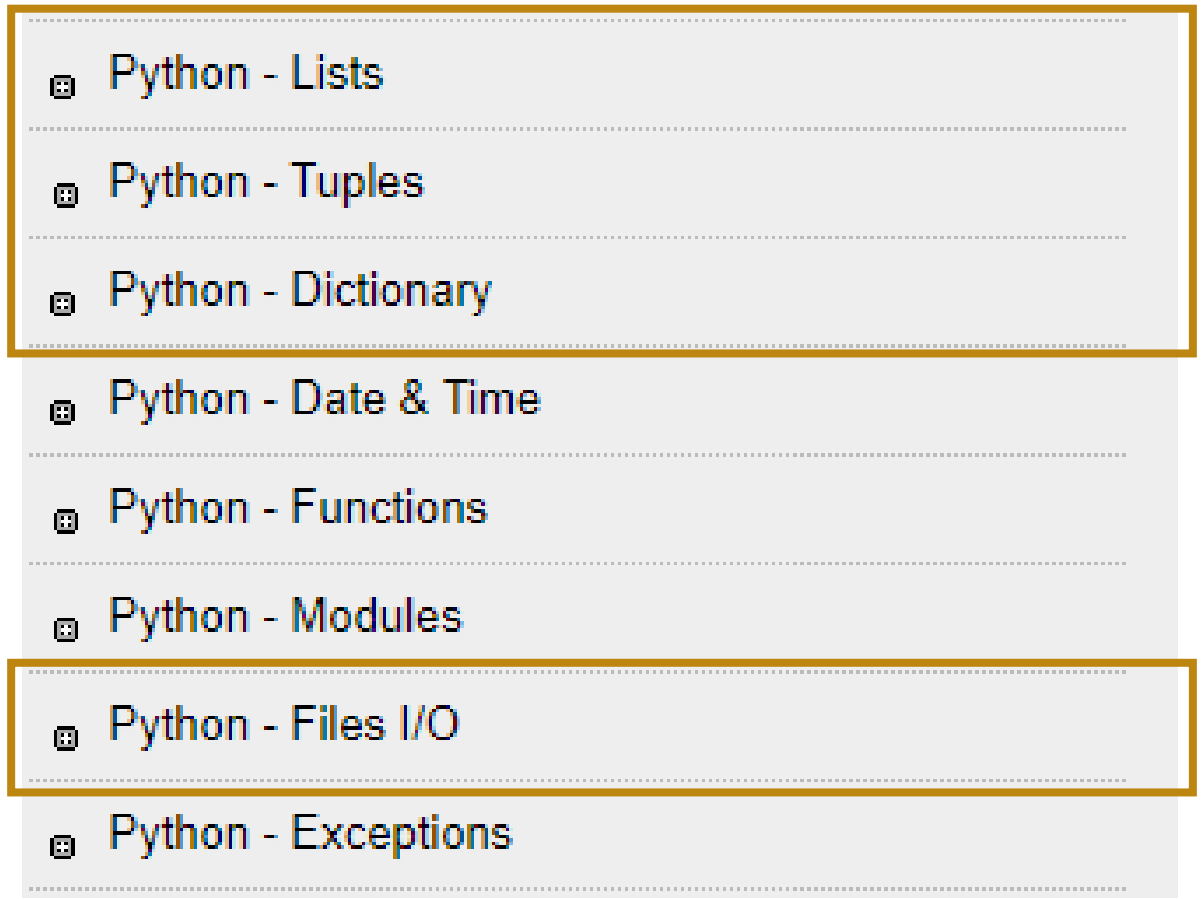
```
>>> 'abc'+123
TypeErrorException: unsupported operand type(s) for +: 'str' and 'int'
>>> 123/0
DivideByZeroException: Attempted to divide by zero.
>>> x = [1, 2, 3]
>>> x[4]
IndexOutOfRangeException: index out of range: 4
>>> x.appenc(4)
MissingMemberException: 'list' object has no attribute 'appenc'
>>> z = x + y
UnboundNameException: name 'y' is not defined
>>> y = {'A':1, 'B':2}
>>> y['a']
KeyNotFoundException: a
>>> u, v = (1, 2, 3)
ValueErrorException: too many values to unpack
```


©2023 ANSYS, Inc. / Confidential

Python基本語法簡介(3/4)

Ansys

https://www.tutorialspoint.com/python/index.htm



©2023 ANSYS, Inc. / Confidential

找出特定目錄底下的特定檔案

- 找出單一目錄底下所有.txt檔。

```
import os
for file in os.listdir("/mydir"):
    if file.endswith(".txt"):
        print(os.path.join("/mydir", file))
```

- 找出單一目錄底下所有.txt檔，包含子目錄。

```
import os
for root, dirs, files in os.walk("/mydir"):
    for file in files:
        if file.endswith(".txt"):
            print(os.path.join(root, file))
```


©2023 ANSYS, Inc. / Confidential

切換工作目錄

- 程式執行時有時需要配置檔或匯入自定義模組，而配置檔往往都是跟程式擺放在同一目錄底下。如不指定目錄，Python會嘗試到工作目錄尋找配置檔，

- 如果Python找不到配置檔，則會引發FileNoFoundException錯誤。為了解決這個問題，建議程式開頭便將工作目錄切換與程式本身所在的目錄。

```
import os

abspath = os.path.abspath(__file__)
dname = os.path.dirname(abspath)
os.chdir(dname)
```


©2023 ANSYS, Inc. / Confidential

import指令

- import指令用來匯入模組
- 自動化常用模組
  - math, os, sys, re, tkinter, matplotlib
- 匯入指令

```
import math
from math import sin, cos
from math import sin as mysin
from math import *
```

©2023 ANSYS, Inc. / Confidential

math module & cmath module

- 基本數學函式庫：https://docs.python.org/3/library/math.html
- 複數函式庫：https://docs.python.org/3/library/cmath.html

©2023 ANSYS, Inc. / Confidential

重點

- Accessing Values in Lists
- Updating Lists
- Delete List Elements
- Basic List Operations
- Indexing, Slicing, and Matrixes
- Built-in List Functions & Methods
- List Comprehension
- Accessing Values in Tuples
- Accessing Values in Dictionary
- Updating Dictionary
- Built-in Dictionary Functions
- Opening and Closing Files
- Reading and Writing Files
- Directories in Python

©2023 ANSYS, Inc. / Confidential

list切片

```
In [7]: x = ['A', 'B', 'C', 'D', 'E', 'F']

In [8]: x[0]
Out[8]: 'A'

In [9]: x[3]
Out[9]: 'D'

In [10]: x[0:3]
Out[10]: ['A', 'B', 'C']

In [11]: x[1:3]
Out[11]: ['B', 'C']

In [12]: x[::2]
Out[12]: ['A', 'C', 'E']
```

```
In [13]: x[1::2]
Out[13]: ['B', 'D', 'F']

In [14]: x[::-1]
Out[14]: ['F', 'E', 'D', 'C', 'B', 'A']

In [15]: x[-2::-2]
Out[15]: ['E', 'C', 'A']

In [16]: x[-3:]
Out[16]: ['D', 'E', 'F']

In [17]: y = x[:]

In [18]: y
Out[18]: ['A', 'B', 'C', 'D', 'E', 'F']
```


©2023 ANSYS, Inc. / Confidential

list運算

宣告空list

x = [ ]

宣告list

x = [4,3,1,5,6,7,2]

加入

x.append(8)

排序

x.sort()

返回list長度

len(x)

返回最大值

max(x)

返回最小值

min(x)

list相加

x + y

元素運算

y = [i*i for i in x]

元素運算+判斷

y = [i for i in x if i%2 == 0]

©2023 ANSYS, Inc. / Confidential

tuple運算

在Python中，Tuple就像是串列（List），不過串列是可變動（Mutable）物件，而Tuple是不可變動（Immutable）物件。你可以使用()來建立Tuple物件，也可以直接逗號區隔元素來建立Tuple物件。

tuple主要用來記錄不同屬性的資料，比方說(name, gender, age)，(id, size, color), (x, y, z)

```
In [1]: x = (1,2,3)

In [2]: x
Out[2]: (1, 2, 3)

In [3]: x = 4,5

In [4]: x
Out[4]: (4, 5)

In [5]: x[0]
Out[5]: 4

In [6]: a, b = x

In [7]: a
Out[7]: 4

In [8]: b
Out[8]: 5
```


©2023 ANSYS, Inc. / Confidential

dictionary運算

- 在 Python 的dictionary當中，每一個元素都由鍵 (key) 和值 (value) 構成，結構為key: value 。不同的元素之間會以逗號分隔，並且以大括號 {}圍住。字典提供了非常快的查詢速度。

```
In [9]: x = {}

In [10]: x
Out[10]: {}

In [11]: x = {'John': ('Male', 23)}

In [12]: x
Out[12]: {'John': ('Male', 23)}

In [13]: x['Mary'] = ('Female', 18)

In [14]: x
Out[14]: {'John': ('Male', 23), 'Mary': ('Female', 18)}

In [15]: x.keys()
Out[15]: dict_keys(['John', 'Mary'])

In [16]: gender, age = x['Mary']

In [17]: gender
Out[17]: 'Female'

In [18]: age
Out[18]: 18
```


©2023 ANSYS, Inc. / Confidential

以頻率對應複數的CSV為例，列舉了幾種不同的資料結構

```
S11_freq=[1,2,3,4,5]
S11_real=[6,7,8,9,10]
S11_imag=[11,12,13,14,15]

S11_a=([1,2,3,4,5], [6,7,8,9,10], [11,12,13,14,15])

S11_b=[(1,6,11),(2,7,12),(3,8,13),(4,9,14),(5,10,15)]

S11_c=[(1, 6+11j), (2, 7+12j), (3, 8+13j), (4, 9+14j), (5, 10+15j)]

S11_d={1:6+11j, 2:7+12j, 3:8+13j, 4:9+14j, 5:10+15j}
```


©2023 ANSYS, Inc. / Confidential

資料結構的選擇

資料結構的選擇沒有絕對的好壞，完全取決於要執行的操作。適合A資料結構的操作對於B資料結構可能相當困難，反之亦然。必要的時候我們可以做資料結構轉換，以適應不同的操作程序。

下面是Python常用於儲存大量資料的資料結構.

- List
- List of tuple
- Tuple of list
- Dictionary

©2023 ANSYS, Inc. / Confidential

w = zip(x , y)

zip可以將多個數值list打包成一個list of tuple，舉例來說，我們將freq, gain和溫度放到list of tuple當中

```
freq = [1e9, 2e9, 3e9, 4e9]
gain = [4, 5, 6, 7]
temp=[30, 25, 20, 18]
data = zip(freq, gain, temp)
```

將數值透過zip關連起來之後，可以容易在for loop當中做篩選處理，比方說，找出滿足gain大於5，溫度小於27所有的頻率點及溫度，可以寫成

```
result=[]
for freq, gain, temp in data:
    if gain>5 and temp<27:
        result.append((freq, temp))
```


©2023 ANSYS, Inc. / Confidential

and x, y = zip(*w)

- 可以透過zip(*)的方法將list of tuple拆成多個list

```
18 freq_list, temp_list = zip(*result)
19 print(freq_list)
20 print(temp_list)
```


©2023 ANSYS, Inc. / Confidential

數字格式化輸出

為了容易閱讀或是要將輸出字串對齊或置中，就可以利用字串的format方法

数字	格式	输出	描述
3.1415926	{:.2f}	3.14	保留小数点后两位
3.1415926	{:+.2f}	+3.14	带符号保留小数点后两位
-1	{:+.2f}	-1.00	带符号保留小数点后两位
2.71828	{:.0f}	3	不带小数
5	{:0>2d}	05	数字补零 (填充左边, 宽度为2)
5	{:x<4d}	5xxx	数字补x (填充右边, 宽度为4)
10	{:x<4d}	10xx	数字补x (填充右边, 宽度为4)
1000000	{:,}	1,000,000	以逗号分隔的数字格式
0.25	{:.2%}	25.00%	百分比格式
1000000000	{:.2e}	1.00e+09	指数记法
13	{:>10d}	13	右对齐 (默认, 宽度为10)
13	{:<10d}	13	左对齐 (宽度为10)
13	{:^10d}	13	中间对齐 (宽度为10)

```
>>>
>>> import math
>>> '{}'.format(math.pi)
'3.1416'
>>> '{:.6f}'.format(math.pi)
'3.141593'
>>> '{:>12.6f}'.format(math.pi)
'    3.141593'
>>> '{:<12.6f}'.format(math.pi)
'3.141593    '
>>> '{:^12.6f}'.format(math.pi)
'  3.141593  '
>>> '{:12.3e}'.format(math.pi)
'   3.142e+00'
>>> '{:+12.3e}'.format(math.pi)
'  +3.142e+00'
>>> '{:+12.3e}'.format(-math.pi)
'  -3.142e+00'
>>> '{:+12.3%}'.format(-math.pi)
'   -314.159%'
>>>
```


©2023 ANSYS, Inc. / Confidential

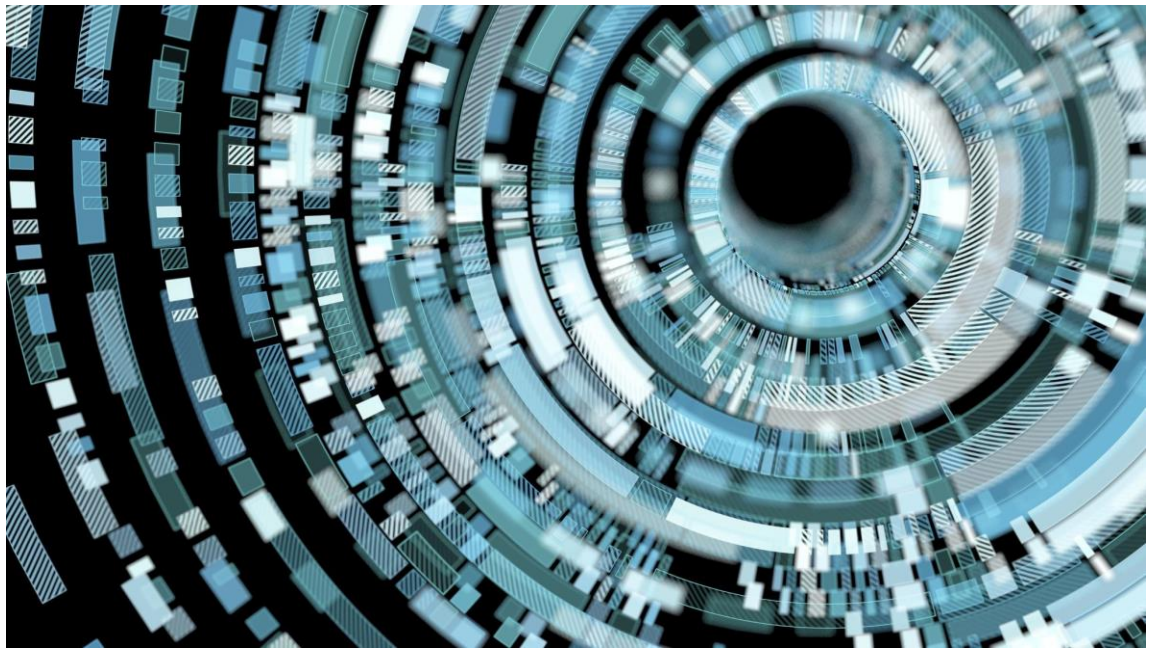
讀/寫文字檔

將每一行讀到list當中

```
with open(file_name) as f:
    text = f.readlines()
```

將list當中的字串寫到檔案當中

```
with open(file_name, ‘w’) as f:
    for i in string_list:
        f.writelines(i + ’\n’)
```



©2023 ANSYS, Inc. / Confidential

例外處理 v.s.事先檢查

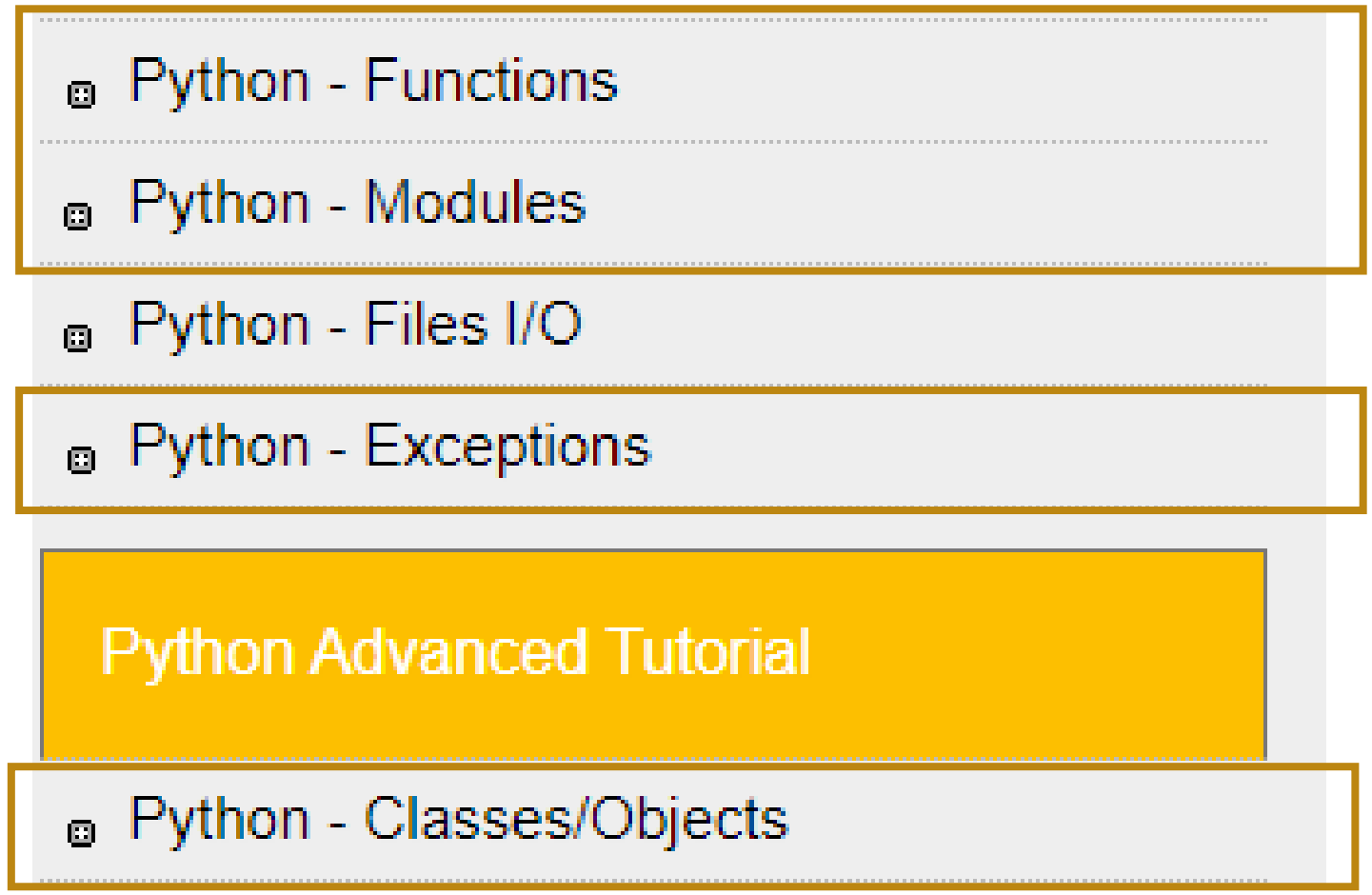
在程式開發階段，一般都是假設輸入參數會嚴格遵守規範，並基於這個假設開發演算法。如果格式出錯，就算這個錯誤無關緊要，程式也會終止運算並返回錯誤訊息，如果要對輸入的資料一一判斷是否合規，將導致程式複雜化並額外耗用運算資源。

一個解決思維是先做再說，發生了正常流程無法處理的狀況，也就是例外，再另行處理。這種設計思維就是所謂的例外處理。目前例外處理已經是主流程式語言所採取的設計方案。這種設計風格又稱之為EAFP(Easier to Ask for Forgiveness than Permission) ,有別於LBYL(Look Before You Leap)。

©2023 ANSYS, Inc. / Confidential

Python基本語法簡介(4/4)

https://www.tutorialspoint.com/python/index.htm



©2023 ANSYS, Inc. / Confidential

重點

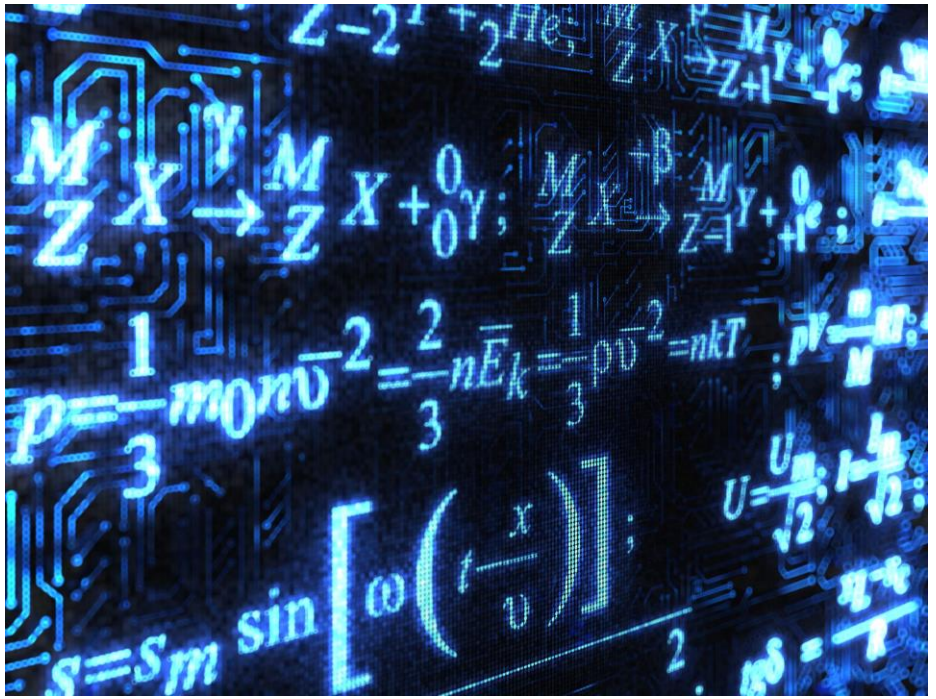
- Defining a Function
  - Scope of Variables
  - Global vs. Local variables
- Calling a Function
  - Pass by reference vs value
  - Function Arguments
  - Keyword arguments
  - Default arguments
- Overview of OOP Terminology
  - Creating Classes
  - Creating Instance Objects
  - Accessing Attributes

©2023 ANSYS, Inc. / Confidential

使用函式的時機為何？

函式將一組完整攻能的代碼包裹起來，僅透過參數傳遞資料，並在完成計算之後回傳結果。其功能主要有下列幾點：

- 需要重複使用某一段程式碼時
- 需要分隔功能，提高程式碼的可讀性時
- 需要限縮變量的範圍時（可簡化變數命名工作）
- 以def定義函式，return返回計算結果，例如：



```
def myadd(x, y):
    return x+y
```


©2023 ANSYS, Inc. / Confidential

函式參數

參數可以是數值，字串，list，tuple, dictionary，函式或是自定義物件。此外參數可以有不同宣告方式：

- Keyword arguments
  - 使用函式時，參數名可連同參數值一同輸入。可提高代碼的可讀性
- Default arguments
  - 不常修改的參數可事先賦予預設值。在呼叫時便可以省去輸入的功夫。

```
>>> def mysub(x, y):
...     return x-y
...
>>> mysub(3,1)
2
>>> mysub(x=3,y=1)
2
>>> mysub(y=1, x=3)
2
>>> def mysub(x, y=1):
...     return x-y
...
>>> mysub(3)
2
>>> mysub(3, 2)
1
>>>
```


©2023 ANSYS, Inc. / Confidential

區域變數與全域變數

- 全域變數定義在函式之外，區域變數定義在函式之內。
- 函式裡面讀取的變數若沒有定義在函數裡面，則會嘗試到函式外部尋找全域變數
- 函式內部如果要修改全域變數需先宣告為global

```
>>> x =100
>>> def foo():
...     print(x)
...
>>> foo()
100
>>> def foo2():
...     x = 30
...     print(x)
...
>>> foo2()
30
>>> x
100
>>> def foo3():
...     global x
...     x =50
...     print(x)
...
>>> foo3()
50
>>> x
50
>>>
```


©2023 ANSYS, Inc. / Confidential

First-Class Citizens

在 Python 中，函數是一個一級公民（first-class citizen）。這意味著，函數與任何其他對象（例如：整數、字符串、列表）一致，既可以動態地創建或銷毀，也可以傳遞給其他函數，或作為值進行返回。

```
>>> import math
>>> def foo(func):
...     return func(math.pi)
...
>>> foo(math.sin)
1.2246063538223773e-16
>>> foo(math.cos)
-1.0
>>>
```

```
>>> for func in [math.sin, math.cos, math.tan]:
...     func(math.pi/6)
...
0.49999999999999994
0.866025403784438 71
0.57735026918962573
>>>
```


©2023 ANSYS, Inc. / Confidential

物件導向設計[WIKI]

物件導向程式設計（英語：Object-oriented programming，縮寫：OOP）是種具有物件概念的程式程式設計典範，同時也是一種程式開發的抽象方針。它可能包含資料、屬性、程式碼與方法。物件則指的是類別的實例。它將物件作為程式的基本單元，將程式和資料封裝其中，以提高軟體的重用性、靈活性和擴充性，物件裡的程式可以存取及經常修改物件相關連的資料。在物件導向程式程式設計裡，電腦程式會被設計成彼此相關的物件。

支援物件導向程式語言通常利用繼承其他類達到代碼重用和可擴展性的特性。而類有兩個主要的概念：

- **類：定義了一件事物的抽象特點。類的定義包含了資料的形式以及對資料的操作。**
- **物件：是類的實例。**

©2023 ANSYS, Inc. / Confidential

Python資料結構都是物件

Python裡面所有的資料結構都是類別，當建立變數時即產生了物件。物件除了可以儲存資料，還可以透過逗號取用當中的值或對應的”**方法**”來對資料作處理，像是：

- x=0.33; y=x.is_integer()
- x=3+4j; y=x.real
- x='abc'; y=x.upper()
- x=[1,2,3]; x.append(4)
- x={'mm':1e-3, 'um':1e-6, 'nm':1e-9}; y=x.keys()

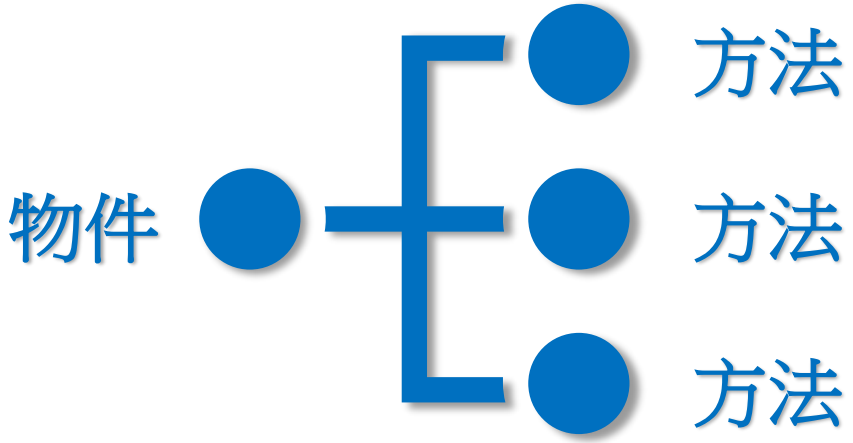
物件可以被建立(Create)，被修改(Update)，被讀取(Read)，被刪除(Delete)。這四個動作簡稱CURD。

©2023 ANSYS, Inc. / Confidential

物件的方法(method)

方法與函數略有不同：

- 方法依附在物件之上
- 方法可以存取物件內的資料



特性：

- 物件可以支援多個方法
- 透過物件名之後加上逗號來使用
- 方法可以返回值或更新物件而不返回值
- 注意的是逗號引用並不表示就是方法，比方說math.sin()是函數，math是模組，不是物件。

©2023 ANSYS, Inc. / Confidential

定義一個類別

```
 7 import math
 8 class vector():
 9     def __init__(self, x, y, z):
10         self.x = x
11         self.y = y
12         self.z = z
13 
14     def __add__(self, other):
15         x = self.x + other.x
16         y = self.y + other.y
17         z = self.z + other.z
18         return vector(x, y, z)
19 
20     def mag(self):
21         return math.sqrt(pow(self.x,2)+pow(self.y, 2)+pow(self.z,2))
22 
23     def __repr__(self):
24         return '({},{},{})'.format(self.x, self.y, self.z)
```


©2023 ANSYS, Inc. / Confidential

用類別宣告物件並做運算

```
26 u = vector(1,2,3)
27 v = vector(3,2,1)
28 w = u + v
29 print(w)
30 z = w + u
31 print(z)
32 print(z.mag())
```

```
In [27]: runfile('C:/Users/mlin/AppData/Roaming/SPB_Data/
untitled7.py', wdir='C:/Users/mlin/AppData/Roaming/SPB_Data')
(4,4,4)
(5,6,7)
10.488088481701515
```


©2023 ANSYS, Inc. / Confidential

魔術方法(進階)

將兩個座標物件相加，我們可以有兩種操作方法：

- Z = X.add(Y)
- Z = X + Y

第二種方法顯然更直覺也更容易輸入。這時候我們可以在類別當中定義__add__(self, other)方法之後，就可以利用運算符號+來操作物件。除了+以外，還有許多的運算符號可以使用，這些統稱魔術方法。右邊顯示的是”部分”的魔術方法。

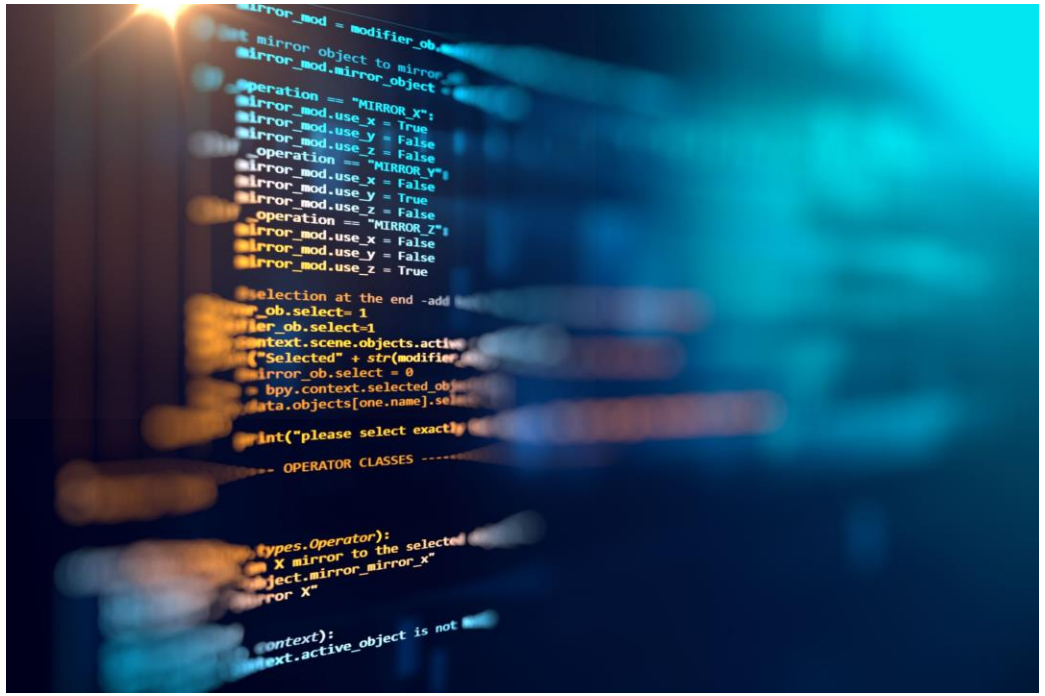
Magic Methods

Python Syntax	Method Call
a + b	a.__add__(b)
a - b	a.__sub__(b)
a * b	a.__mul__(b)
a / b	a.__truediv__(b)
a // b	a.__floordiv__(b)
a % b	a.__mod__(b)
a ** b	a.__pow__(b)
a == b	a.__eq__(b)
a != b	a.__ne__(b)
a < b	a.__lt__(b)
a > b	a.__gt__(b)
a <= b	a.__le__(b)
a >= b	a.__ge__(b)

©2023 ANSYS, Inc. / Confidential

PEP8 PYTHON 編碼規範手冊

PEP8 是 Python 社群共通的風格指南，一開始是 Python 之父 Guido van Rossum 自己的撰碼風格，慢慢後來演變至今，目的在於幫助開發者寫出可讀性高且風格一致的程式。許多開源計畫，例如 Django 、OpenStack等都是以 PEP8 為基礎再加上自己的風格建議。



©2023 ANSYS, Inc. / Confidential


Ansys