

# **Ansys Notebook Documentation**

*Release (Limited Preview)*

**Ansys, Inc.**

May 14, 2024



<b>List of Tables</b>	<b>iii</b>
<b>1 Introduction and Purpose</b>	<b>1</b>
1.1 User's Guide . . . . .	1
1.2 Features . . . . .	10
1.3 Tutorials . . . . .	26
1.4 Issues . . . . .	26
1.5 Support . . . . .	27



Table 1.1: Pod Compute Resource Limits. . . . . 27



---

# Introduction and Purpose

---

Welcome to the documentation for Ansys Notebook. Here, you will find materials to help you use Ansys Notebook, our cloud-native interactive programming experience for simulation.

## 1.1 User's Guide

---

**Note** Limited Preview version.

---

### 1.1.1 Overview

Ansys Notebook is a cloud-native interactive programming experience for simulation. Conveniently hosted within the OnScale Solve application, Notebook provides a cell-based python execution environment and PyAnsys client library support. User notebooks are part of the OnScale Solve Project dashboard and leverage the same workflows and user experience as OnScale Portal, Solve, and Modeler.

### 1.1.2 Getting Started

#### Pre-Requisites

- Access to a public internet connection.
- Access to a web browser.
- A verified user account on [OnScale Solve](#).

---

**Note** Notebook is a Limited Preview, access is by an approved basis only. Please contact your Ansys account manager for more information.

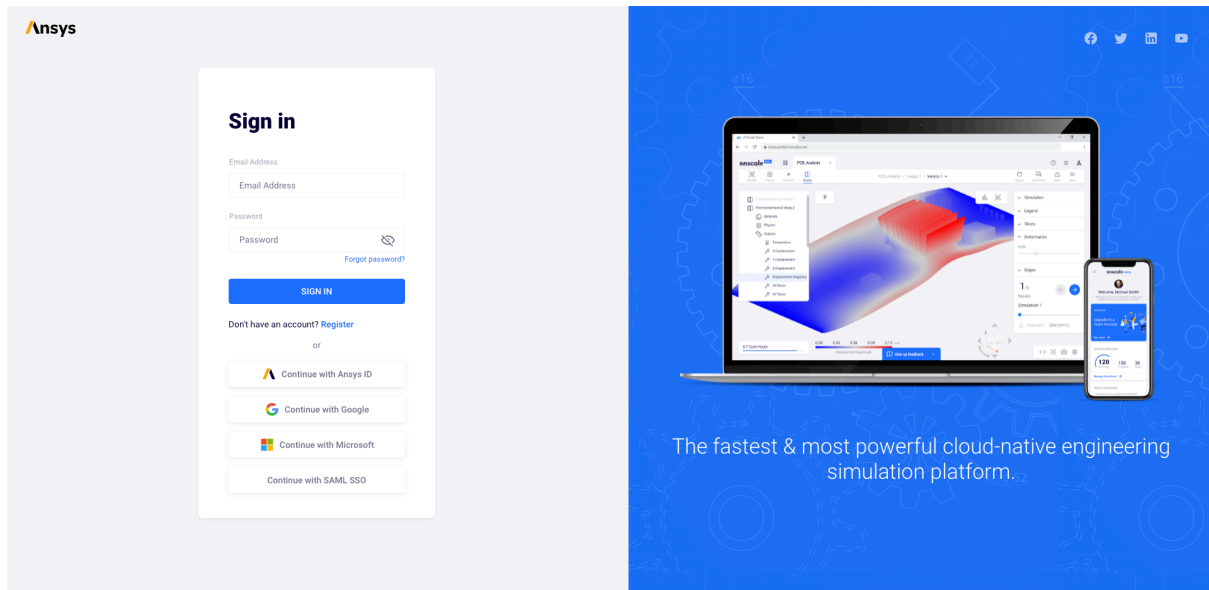
---

#### Interacting with Notebook

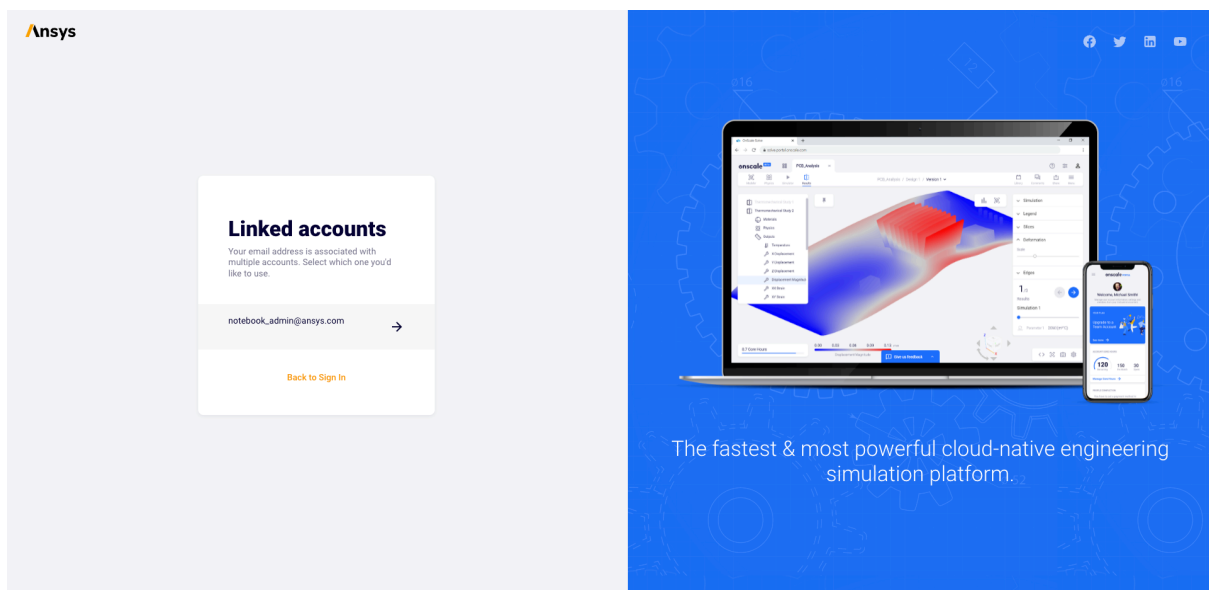
Notebook as a product feature is accessed directly within OnScale Solve.

##### *Logging in*

- Point your browser to <https://solve.prod.portal.onscale.com>.

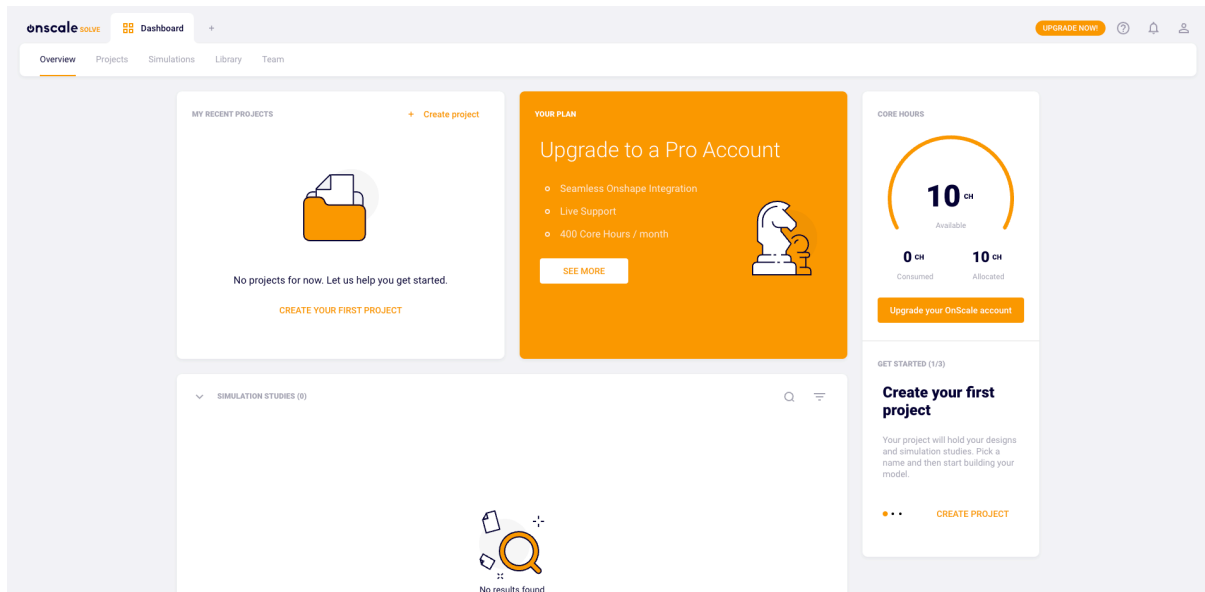


- Enter your username and password.
- Select the associated Notebook-enabled account.



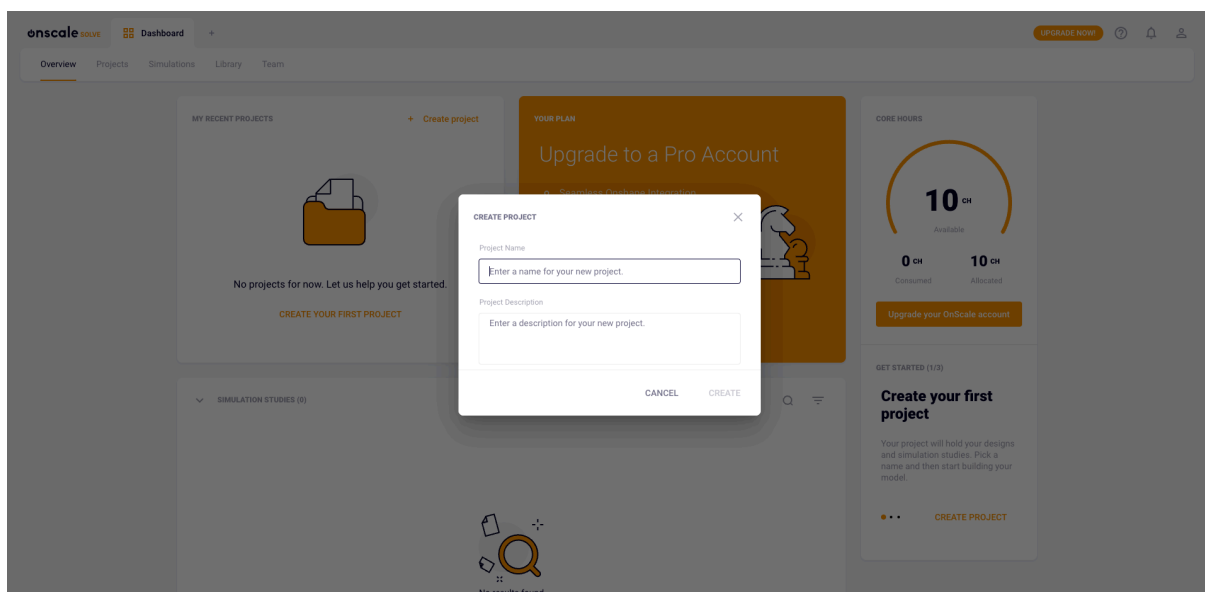
- You will be redirected to your Solve dashboard.



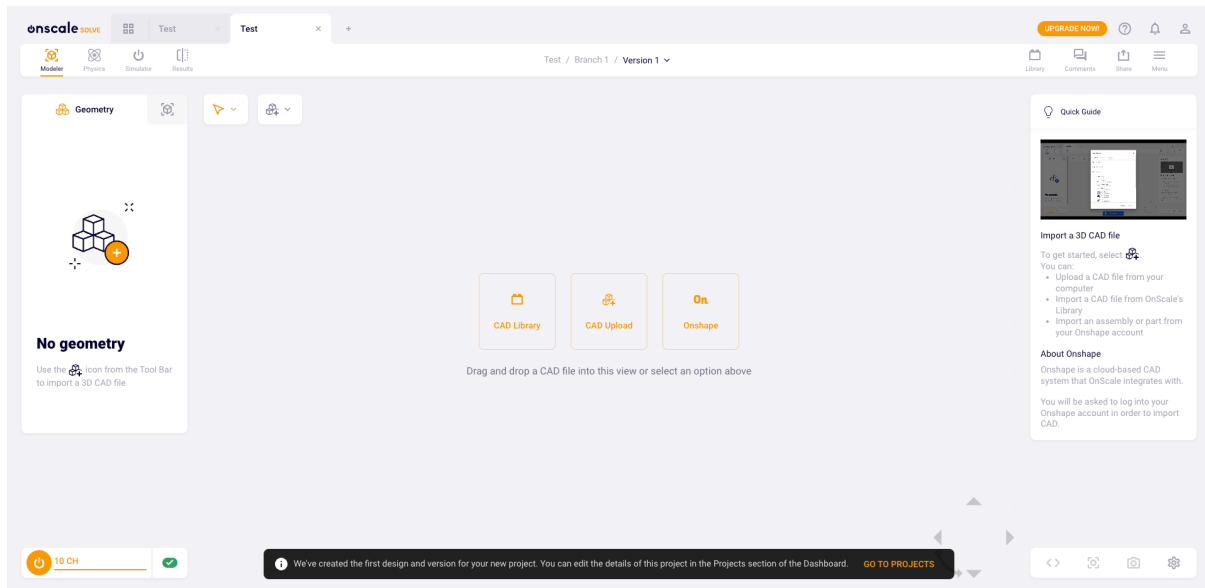


### Creating a Project

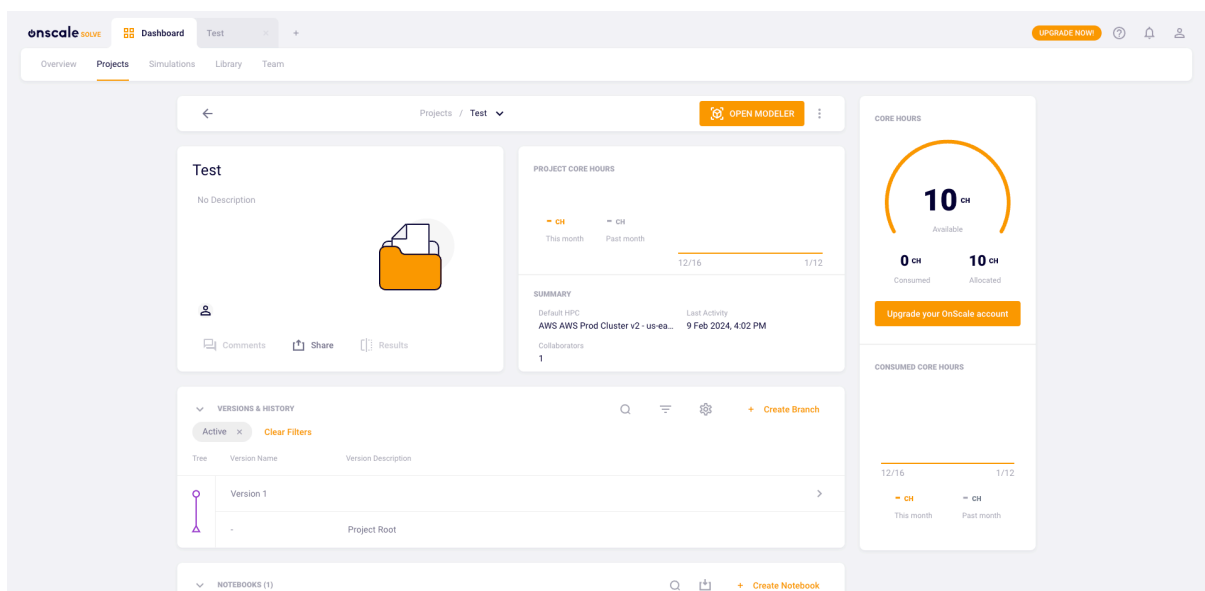
- Click on *Create project* from the top left tile.
- Enter a Project name and description:



- Click on the “Go To Projects” toast message at the bottom of the screen:

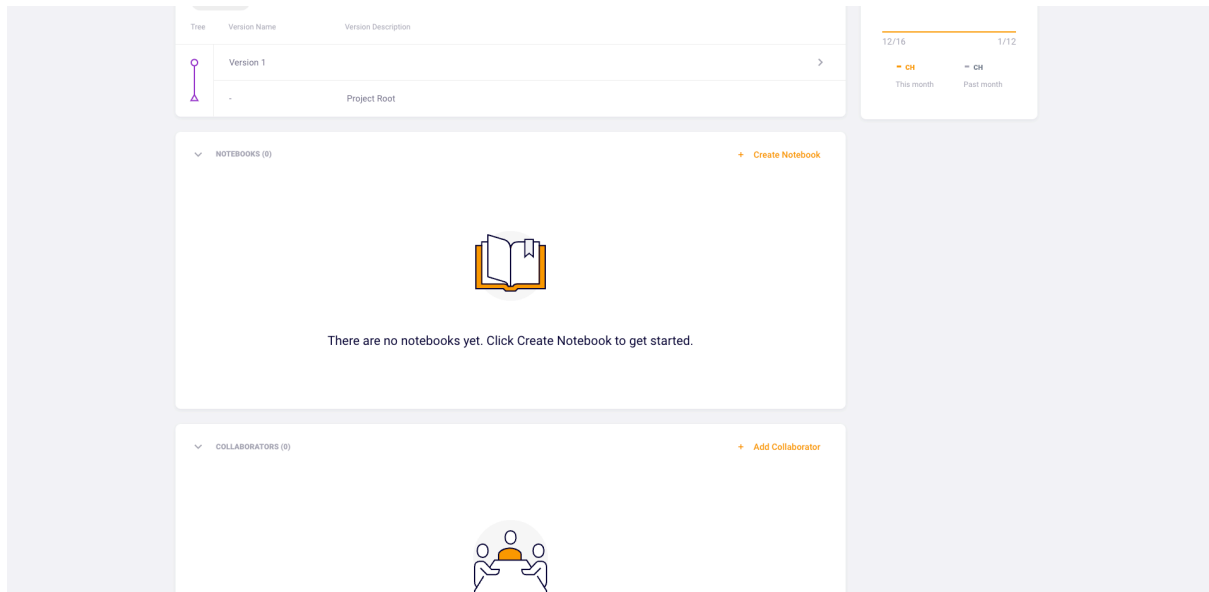


- You will arrive at the Project Dashboard:

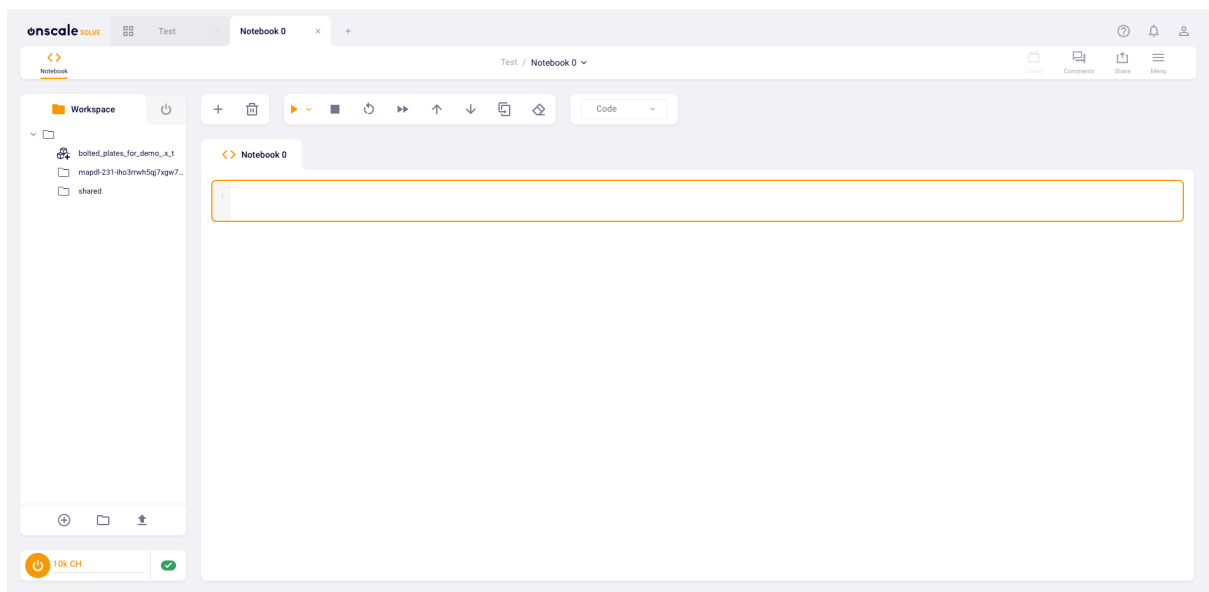


### Creating a Notebook

- From the project dashboard, scroll down to the *Notebooks* section:

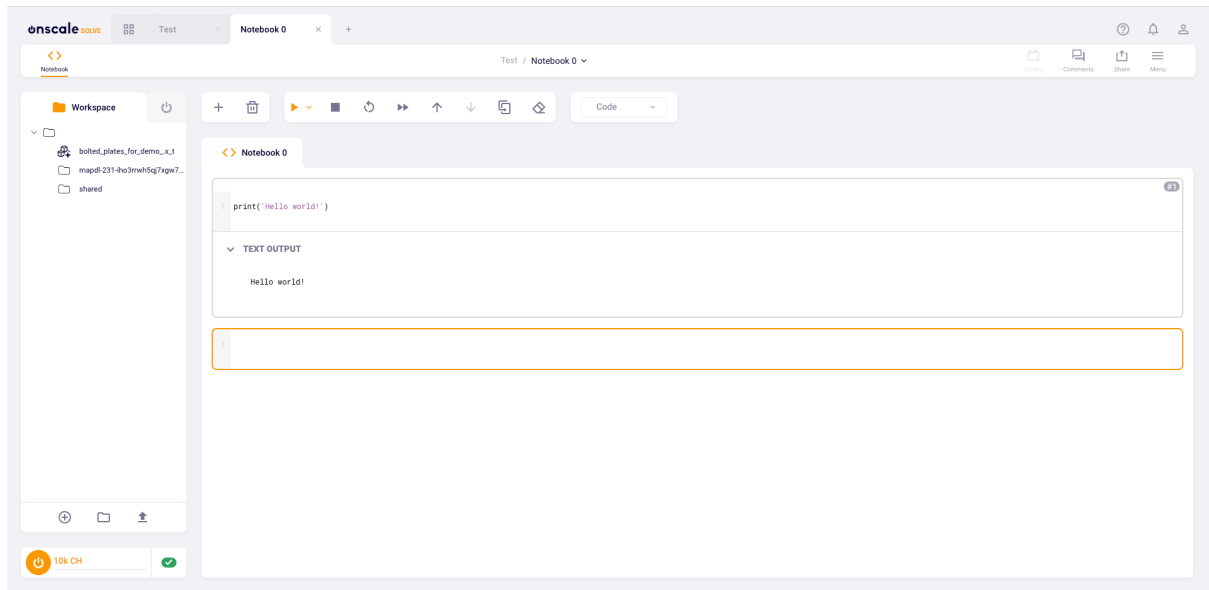


- Click on the “Create Notebook” link.
- A new notebook will be created and you will be redirected to the new notebook:



### Working inside a Notebook

- You can enter python snippets in your notebook and execute them via the *Play* button:



### Interacting with Ansys Products

- You can interact with Ansys products via the PyAnsys client libraries. For example, a simple PyMAPDL workflow:
- Start by importing the PyMAPDL client library and launching an instance:

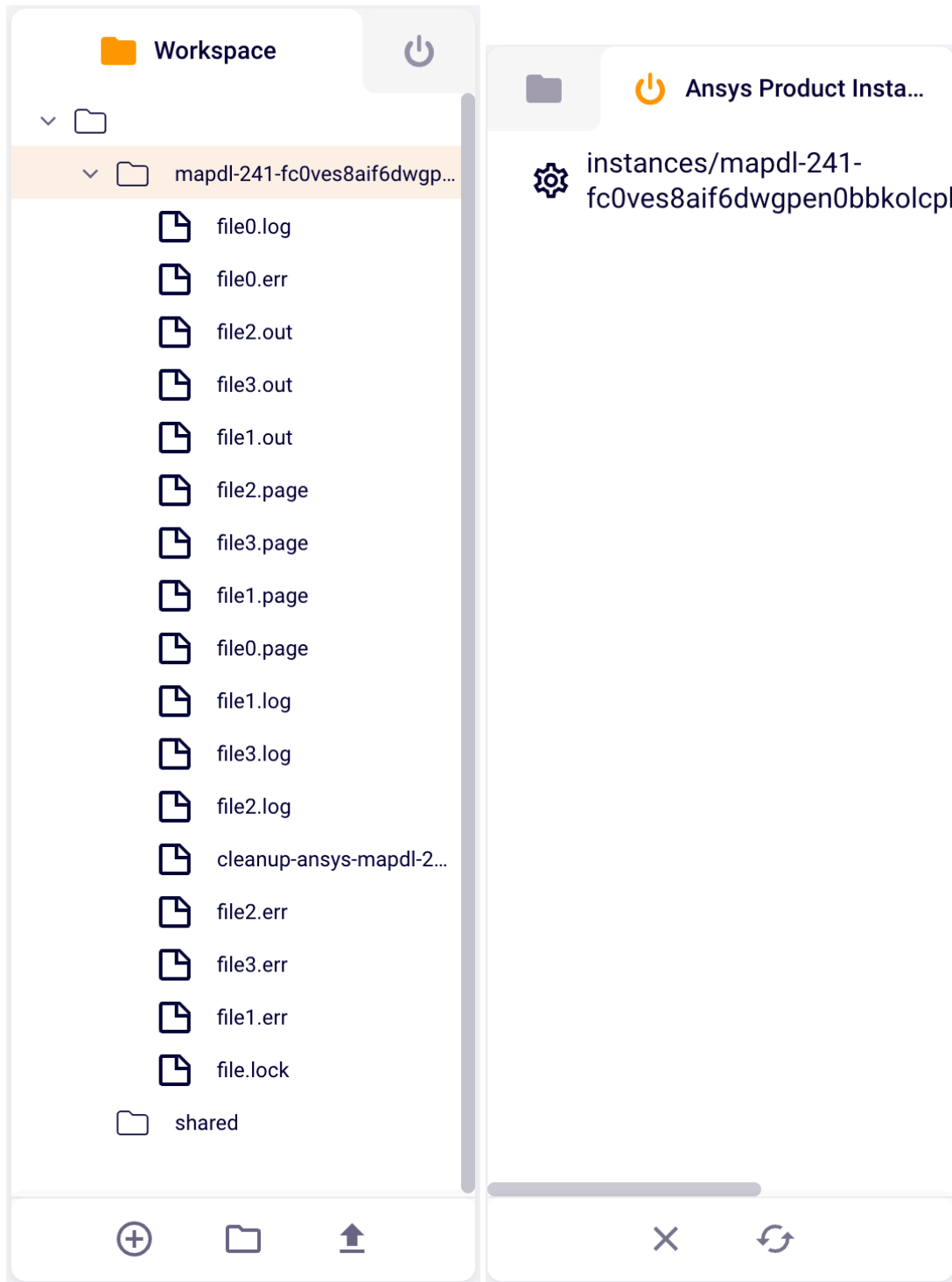
```
from ansys.mapdl.core import launch_mapdl
mapdl = launch_mapdl()
```

#### PyMAPDL Example

##### Create MAPDL Instance

```
1 from ansys.mapdl.core import launch_mapdl
2 mapdl = launch_mapdl()
```

- When the product has been launched, you can see product files in the workspace on the left. You can also see the product in the product instance panel.



- After the product is running, you can setup the model and mesh:

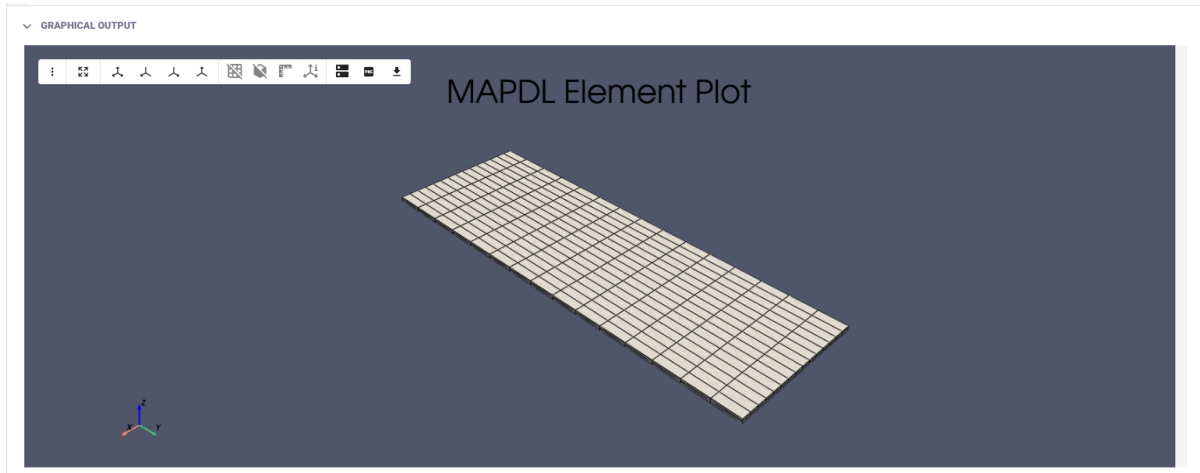
```
mapdl.prep7()
mapdl.mp('kxx', 1, 45)
mapdl.et(1, 90)
mapdl.block(-0.3, 0.3, -0.46, 1.34, -0.2, -0.2 + 0.02)
mapdl.vswEEP(1)
p = mapdl.eplot(return_plotter=True)
p.show()
```

### Set Up Model and Mesh

```
1 mapdl.prep7()
2 mapdl.mp('kxx', 1, 45)
3 mapdl.et(1, 99)
4 mapdl.brick(-0.3, 0.3, -0.45, 1.34, -0.2, -0.2 + 0.02)
5 mapdl.vswEEP(1)
6 p = mapdl.eplot(return_plotter=True)
7 p.show(jupyter_backend='tornado')
```

> GRAPHICAL OUTPUT

- You can interact with the output in the *Graphical Output* window:



- Apply constraints for the simulation:

```
mapdl.asel('S', vmin=3)
mapdl.nsla()
mapdl.d('all', 'temp', 5)
mapdl.asel('S', vmin=4)
mapdl.nsla()
mapdl.d('all', 'temp', 100)
out = mapdl.allsel()
mapdl.vswEEP(1)
```

### Set up Constraints

```
1 mapdl.asel('S', vmin=3)
2 mapdl.nsla()
3 mapdl.d('all', 'temp', 5)
4 mapdl.asel('S', vmin=4)
5 mapdl.nsla()
6 mapdl.d('all', 'temp', 100)
7 out = mapdl.allsel()
8 mapdl.vswEEP(1)
```

- Run the solve for the model:

```
mapdl.run('/SOLU')
print(mapdl.solve())
out = mapdl.finish()
```

### Run the Solve

```
1 mapdl.run('/SOLU')
2 print(mapdl.solve())
3 out = mapdl.finish()
```

> TEXT OUTPUT

- Visualize the results:

## Preview)

```
mapdl.post1()
mapdl.set(1, 1)
p = mapdl.post_processing.plot_nodal_temperature(return_plotter=True)
p.show()
```

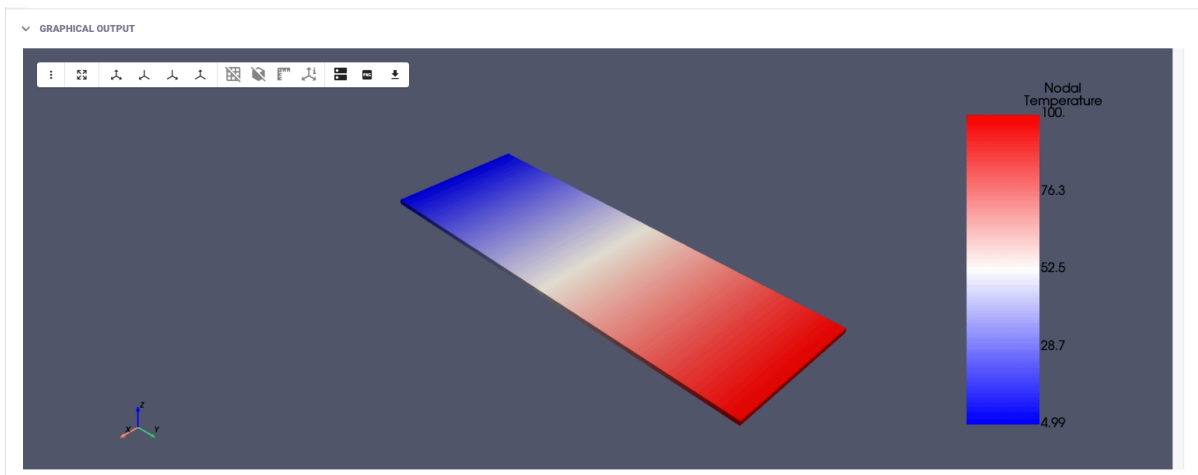
### Process the Results (Visually)

```
mapdl.post1()
mapdl.set(1, 1)
p = mapdl.post_processing.plot_nodal_temperature(return_plotter=True)
p.show(jupyter_backend='trame')
```

> GRAPHICAL OUTPUT

- Here is what the *Graphical Output* should look like:

**Note** By default PyVista uses client-side rendering. Use `plot.show(jupyter_backend='trame')` to display the plot with server-side rendering. There will be minor lag when changing to server-side rendering.



- Alternatively, you can interact with the results programmatically:

```
result = mapdl.result
nnum, temp = result.nodal_temperature(0)
print(nnum)
```

### Process the Results (Programmatically)

```
result = mapdl.result
nnum, temp = result.nodal_temperature(0)
print(nnum)
```

> TEXT OUTPUT

```
[ 1  2  3 ... 2718 2719 2720]
```

- Finally, view the nodal results using pandas:

```
import pandas as pd
d = pd.DataFrame({'Node Number': nnum, 'Temperature': temp}, columns=['Node Number', 'Temperature'])
print(d)
```

Bring the Nodal Results in to Pandas

```
1 import pandas as pd
2 d = pd.DataFrame({'Node Number': nnum, 'Temperature': temp}, columns=['Node Number', 'Temperature'])
```

```
print(d)
```

TEXT OUTPUT

	Node Number	Temperature
0	1	11.413421
1	2	17.709608
2	3	24.834726
3	4	30.359331
4	5	36.685264
...	...	...
2715	2716	96.783801
2716	2717	96.781387
2717	2718	96.777535
2718	2719	96.768937
2719	2720	96.733719

[2720 rows x 2 columns]

- When you are ready to end your session, you can terminate the product instance:

```
mapdl.exit()
```

Terminate the MAPDL instance

```
mapdl.exit()
```

End of Example

You have completed the Getting Started chapter!

To learn more about Ansys Notebook, please review the *Features* chapter.

## 1.2 Features

---

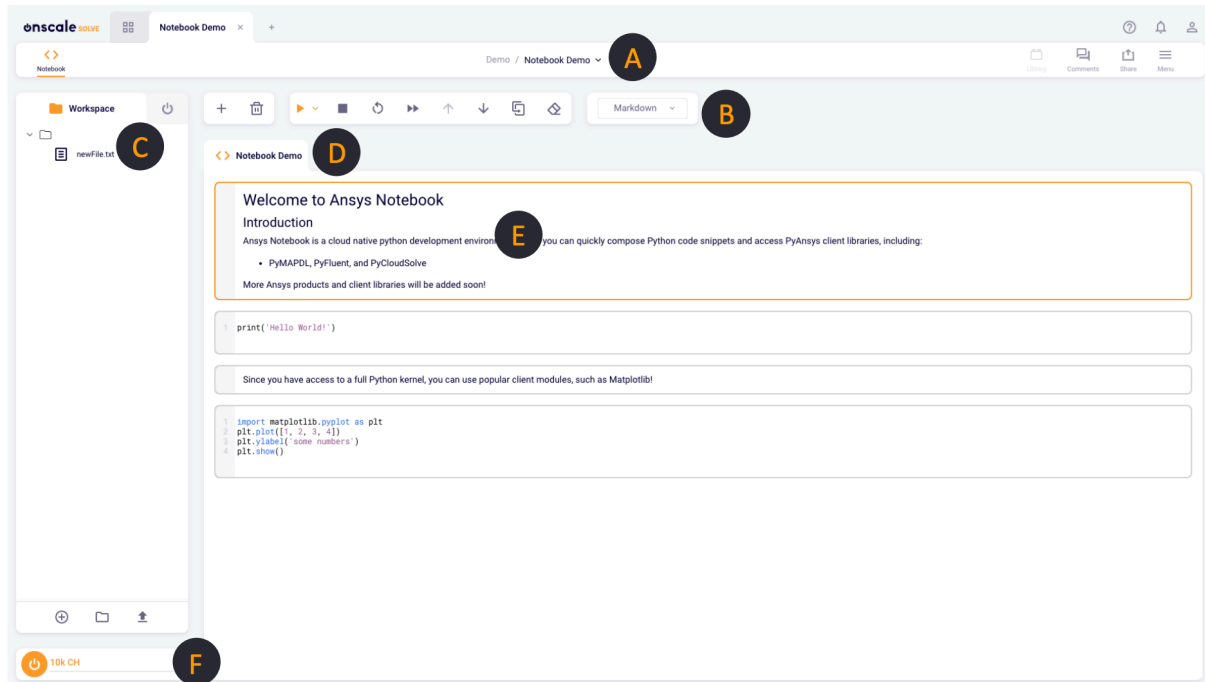
**Note** Limited Preview version.

---

This chapter introduces the key Notebook features you will encounter with regular use.



## 1.2.1 User Interface



The Notebook user interface consists of the following components:

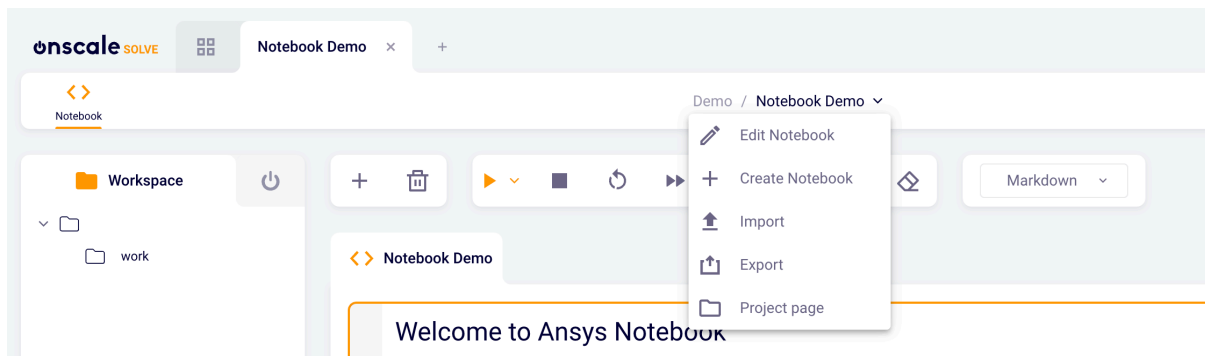
1. Navigation Bar
2. Actions
3. Session Manager
4. Notebook Tabs
5. Cells
6. Status

### Navigation Bar







The Navigation Bar allows you to switch between top-level tabs, including the Project Dashboard, Modeler, and other Notebooks.

You can also access the quick-edit drop down menu from the Notebook name:



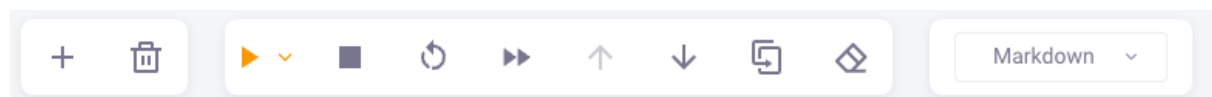
Available actions include:




-  Edit Notebook  
**Edit Notebook**  
 Edit the current Notebook (Rename, Delete)
-  Create Notebook  
**Create Notebook**  
 Create a new Notebook
-  Import  
**Import**  
 Import a new Notebook (Jupyter Notebook | .ipynb)
-  Export  
**Export**  
 Export the current Notebook (Python | .py , Text | .txt , or Jupyter Notebook | .ipynb)
-  Project page  
**Project Page**  
 Return to Project Dashboard

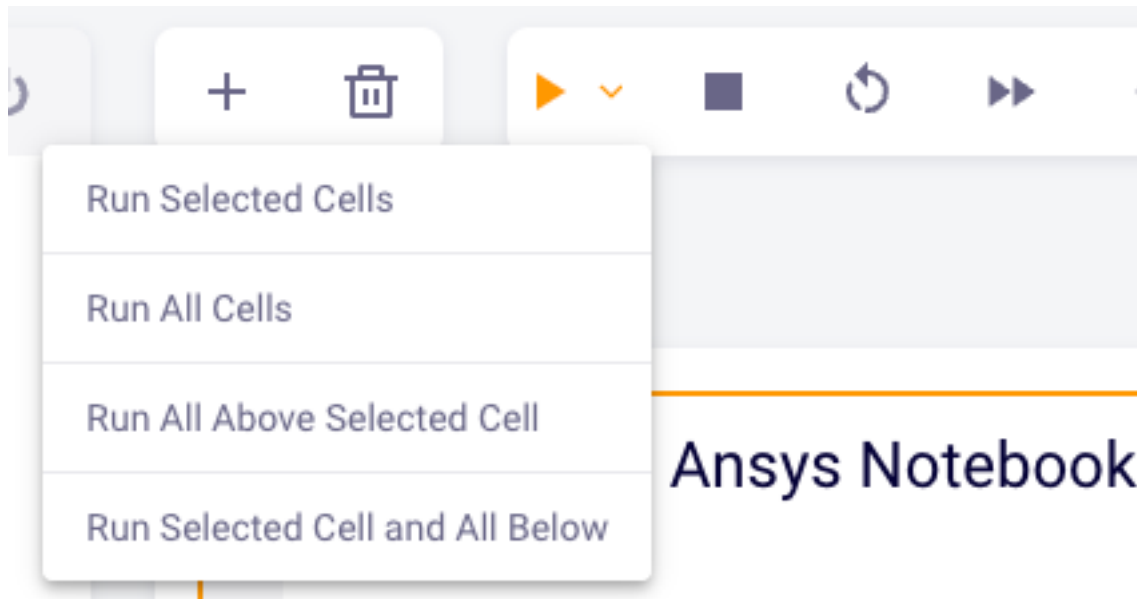
## Actions

The Notebook Actions area provides all functionality to efficiently manage your notebook's cells.

**Note** Most actions require you to select a cell. To select a cell, click anywhere within the border of a cell. You can identify a selected cell by viewing a blue-highlight around the cell border.



- 
**Add Cell**  
 Inserts a new cell below the currently selected cell.
- 
**Delete Cell**  
 Deletes the currently selected cell.
- 
**Execute Cell**  
 Provides options to execute a cell:



- *Run Selected Cell*  
Executes the content of the currently selected cell and immediately stops, selecting the next cell.

---

**Note** You can also use the keyboard shortcut *SHIFT + ENTER* to execute the currently selected cell.

---

- *Run All Cells*  
Executes the content of all cells in the notebook, starting at the first cell and continuing in sequence until the end of the notebook.
- *Run All Above Selected Cell*  
Executes the content of all preceding cells. This is convenient when the preceding cells perform required setup before the current cell can execute.
- *Run Selected Cell and All Below*  
Executes the content of the currently selected cell and all following cells.

- **Interrupt**  
Stops the current cell execution.



- **Restart**  
Restarts the python kernel and clears the cell outputs for the current notebook. This is useful when you want a clean environment with no artifacts from past cell executions.



- **Fast-Forward**  
Restarts the python kernel and executes all cells within the current notebook.





- **Move Up**  
Moves a cell before the preceeding cell.

---

**Note** You can also interactively drag-and-drop a cell to a new position.

---



- **Move Down**  
Moves a cell after the following cell.

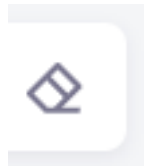
---

**Note** You can also interactively drag-and-drop a cell to a new position.

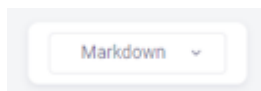
---



- **Duplicate**  
Duplicates the currently selected cell and inserts the new cell immediately after the selected cell.

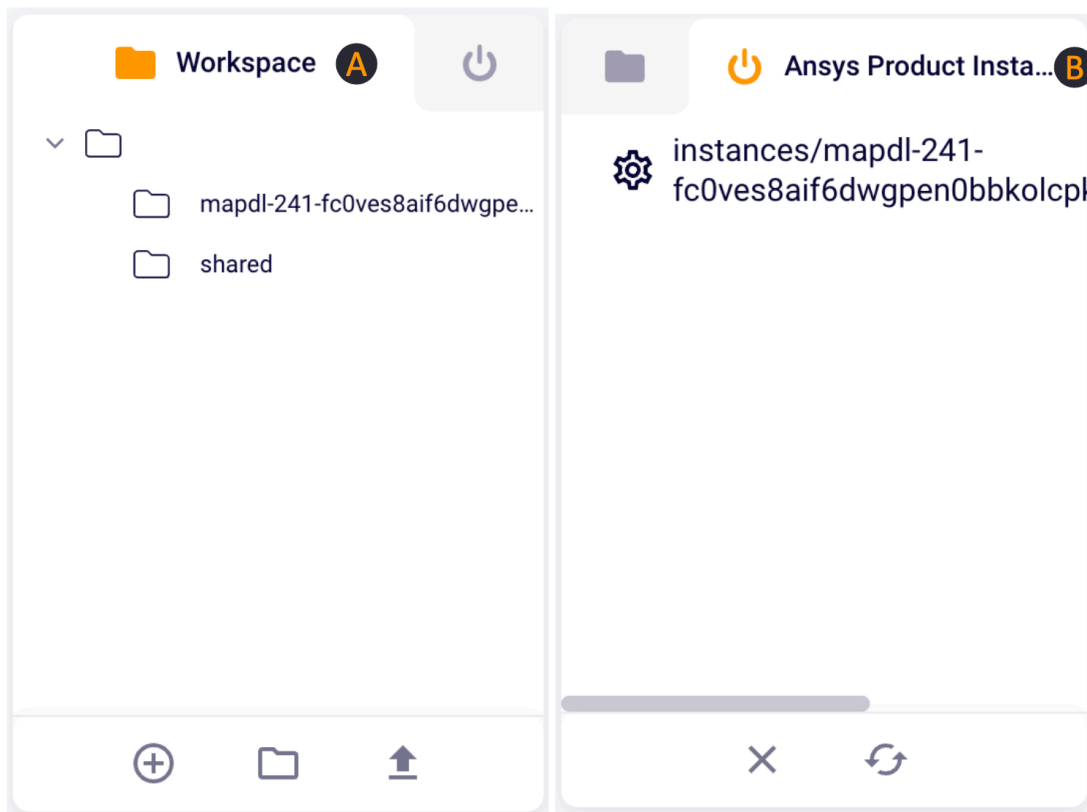


- **Clear All Outputs**  
Clears all cell outputs for the current notebook.



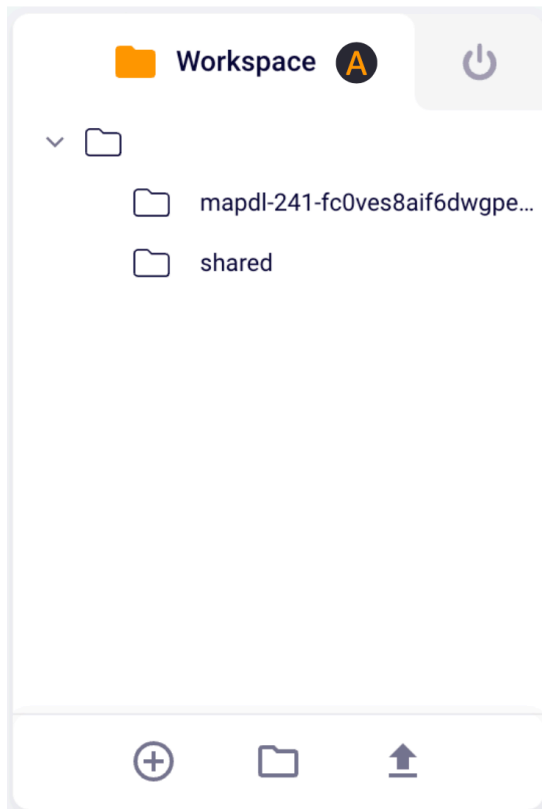
- **Cell Type**  
Sets the cell content type for the currently selected cell. Supported types:
  - *Code*  
Python code with syntax highlighting. When you execute a cell of this type, the content is sent for processing to the Python kernel.
  - *Markdown*  
Markdown syntax. When you execute a cell of this type, the content is rendered to its HTML equivalent.

### Session Manager



The Session Manager provides basic file administration and product instance management for your notebook. The Manager is split into two main components:

1. Workspace
2. Ansys Product Instances

*Workspace*

The Workspace provides persistent file management for your notebook. The Workspace is a convenient location to store files that you want to use with your notebook or share with others. When a new product instance is launched, the associated product files will populate under a new folder in the Workspace. This folder will be named according to the product in the product instance tab. You can perform the following actions with the Workspace:

- Move files from your local disk to Notebook for processing.
- Move files from your Notebook session to your local disk.
- Share files with others by using the `/shared` folder.

For example, you can upload a CAD file to your Workspace. You can read the file into an Ansys product running in your Notebook session. The Ansys product may generate files as output. You can optionally download these files from the Workspace for post processing or alternate storage.

---

**Note** The Workspace performs periodic refreshes to display up-to-date item status. However, you may experience a delay. To speed up the refresh sequence, you can click between the Workspace and Product Instance tabs.

---

---

**Note Do not** delete product related Workspace files while the product instance is running. This may cause the product instance to crash.

---

---

**Note** IPYNB files are **not** supported in the Workspace. All other file types are supported.

---

---

**Note** Workspace files and directories are sorted by their creation date, with the most recently created items appearing first. Currently, this default sorting order cannot be modified.

---

Available Workspace Actions:

**Note** The three main Workspace actions require that you first click on the target item in the Workspace before performing the action. For example, to upload a file, first select the destination folder in the Workspace. Then proceed to execute the upload file action.



- **New File**

Creates an empty file in your Workspace.



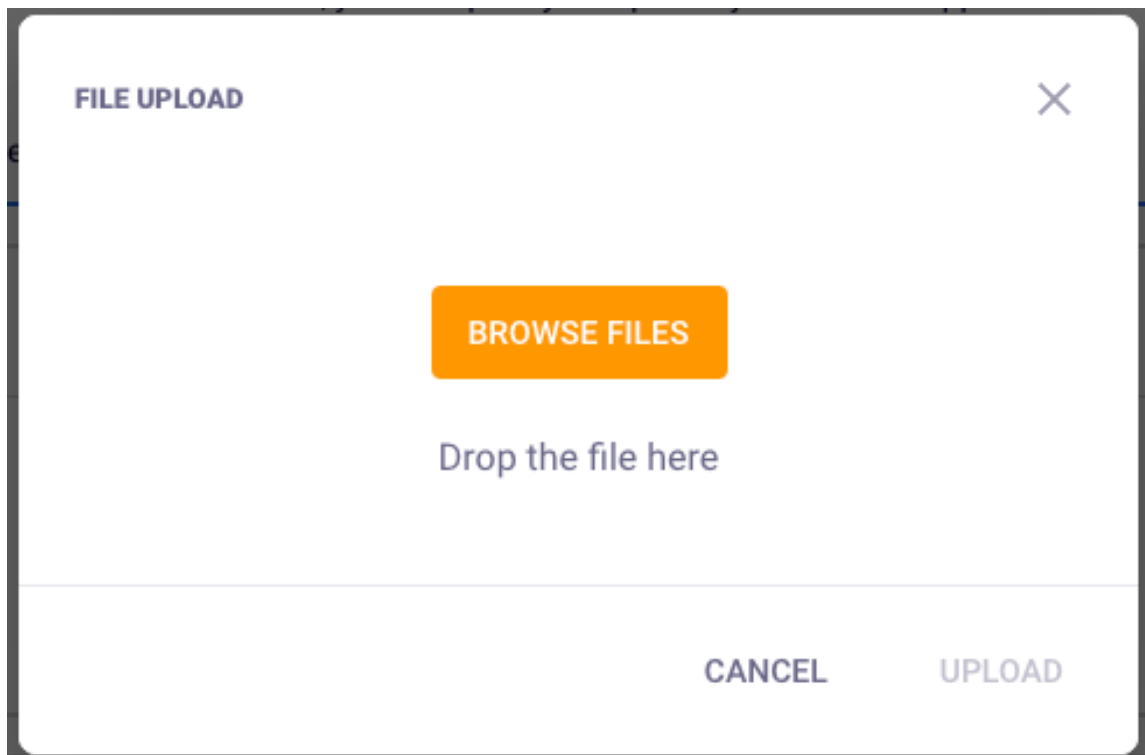
- **New Folder**

Creates an empty folder in your Workspace.



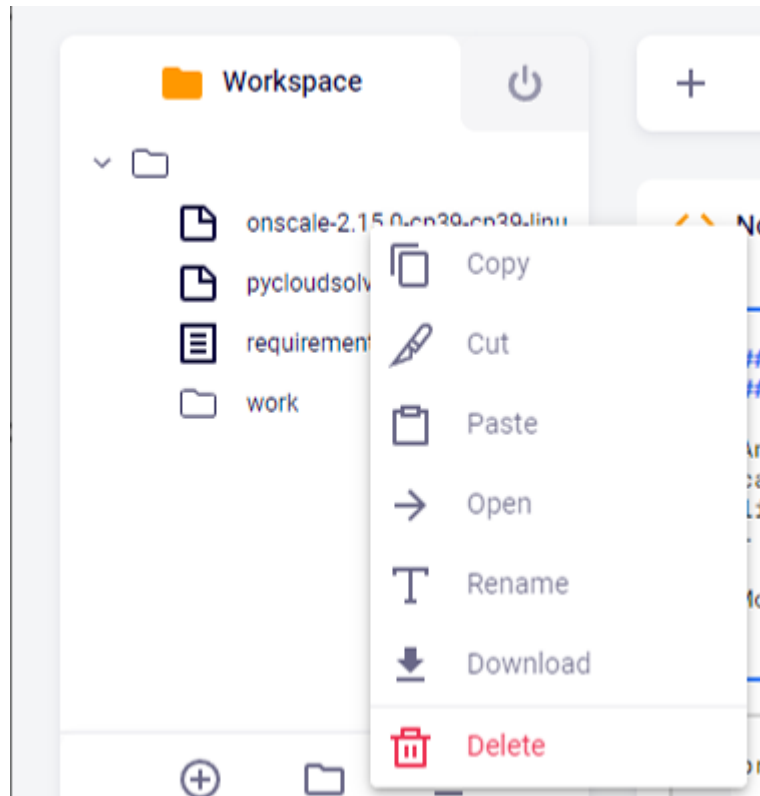
- **Upload File**

Uploads a file from your local disk to your Workspace. You will be prompted by a second screen to select your file:







Workspace Item Actions:

You can right-mouse-button click on a folder or file in your Workspace to access the context menu:



Sub-actions include:

-  **Copy**  
Places the item into a clipboard for pasting elsewhere in the Workspace.
-  **Cut**  
Removes the item from the workspace and places it in a clipboard for pasting elsewhere in the Workspace.
-  **Paste**  
Inserts the active clipboard item into the selected Workspace location.
-  **Open**  
Opens the selected Workspace file for editing in the Notebook Tab area.




---

**Note** You can also double-click on a file in the Workspace to open the file for editing.

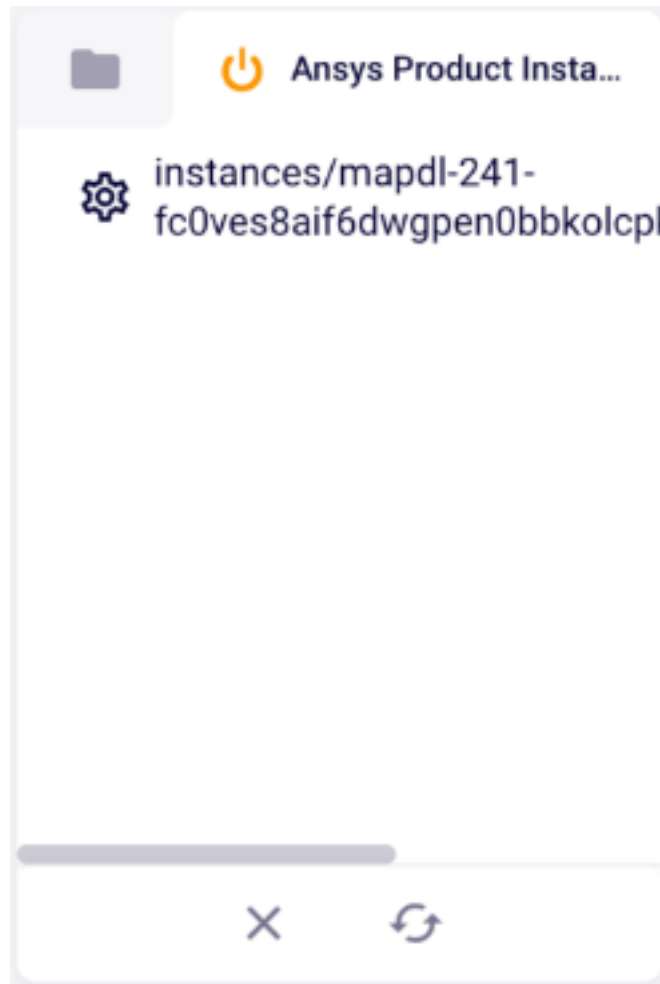
---



Preview)

-  **Rename**  
 Edits the Workspace item name.
-  **Download**  
 Downloads the Workspace item to your local disk.
-  **Delete**  
 Permanently removes the item from the Workspace.

*Ansys Product Instances*



The Ansys Product Instances area provides instance management for your notebook.

When you create an product instance by using a PyAnsys client library, a new entry will appear in the instances list.

Available actions include:



- **Shut Down All Product Instances**  
Terminates all running product instances displayed in the list.

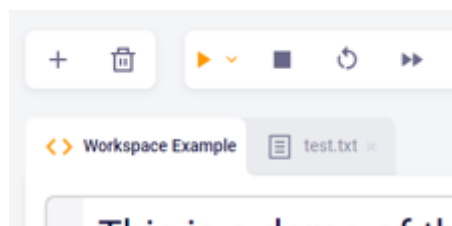


- **Refresh Product Instance List**  
Updates the product instance list to report the latest instance status.

You can also click the *x* on a specific product instance in the list to shutdown only that instance:



## Notebook Tabs



The Tab Area of Notebook provides convenient navigation between your current notebook cells and additional Workspace files.

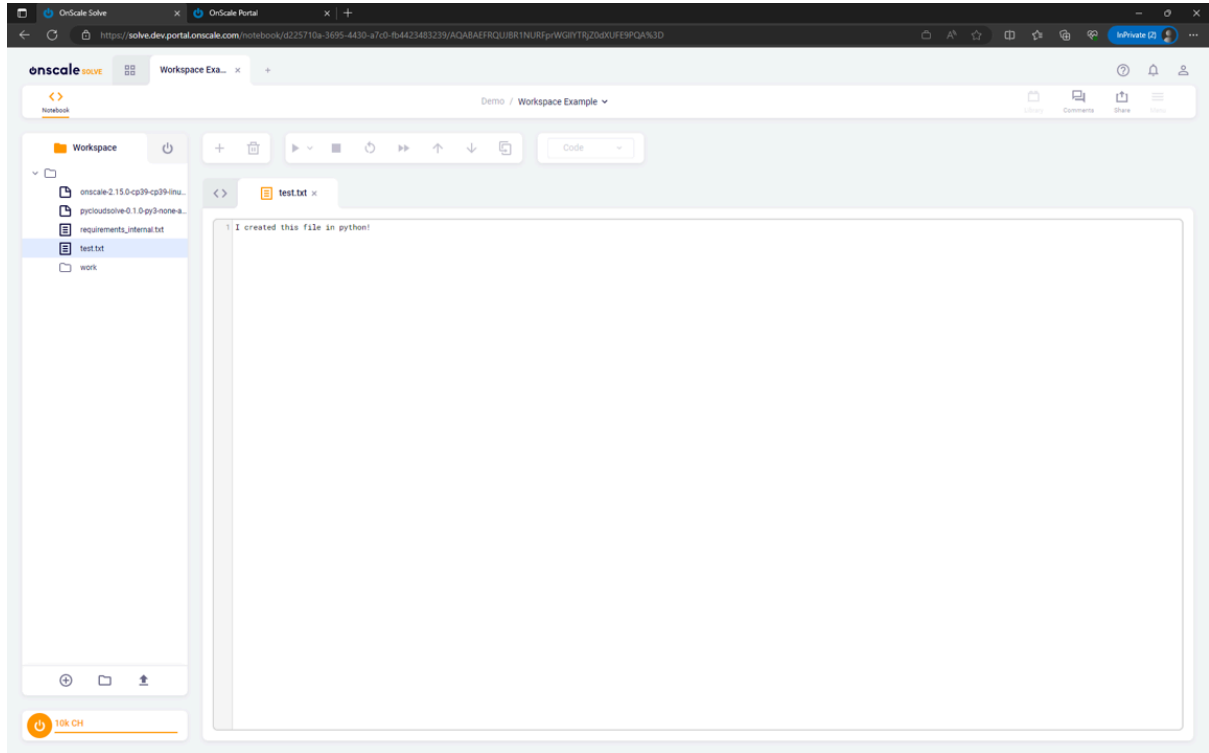
By default, the Tab Area displays one entry: the active notebook.

---

**Note** You can not close the active notebook tab.

---

When you interact with a file in the Workspace (either by double-click or right-click-open actions) a new tab will appear to support file viewing and editing:



## Cells

### PyFluent Demo

#### Watertight geometry meshing workflow

This example sets up and solves a three-dimensional turbulent fluid flow and heat transfer problem in a mixing elbow, which is common in piping systems in power plants and process industries. Predicting the flow field and temperature field in the area of the mixing region is important to designing the junction properly.

This example uses the guided workflow for watertight geometry meshing because it is appropriate for geometries that can have no imperfections, such as gaps and leakages.

##### Workflow tasks

The watertight geometry meshing workflow guides you through these tasks:

- Import a CAD geometry
- Generate a surface mesh
- Describe the geometry
- Generate a volume mesh

##### Problem description

A cold fluid at 20 deg C flows into the pipe through a large inlet. It then mixes with a warmer fluid at 40 deg C that enters through a smaller inlet located at the elbow. The pipe dimensions are in inches, and the fluid properties and boundary conditions are given in SI units. Because the Reynolds number for the flow at the larger inlet is 58, 888, a turbulent flow model is required.

#### Example Setup

Before you can use the watertight geometry meshing workflow, you must set up the example and initialize this workflow.

##### Perform required imports

Perform required imports, which includes downloading and importing the geometry file.

```
1 import ansys.fluent.core as pyfluent
2 from ansys.fluent.core import examples
3 import pyvista as pv
```

The main Notebook area, Cells allow you to add code and markdown to your notebook. By default, a notebook has one cell. You can add additional cells on-the-fly through the *add cell* action.

Each cell is made up of an input area:

```
print('Hello World!')
```

Inside the input area, you can press *TAB* to auto-complete code.

```
1 pyfluent.
```

```
fx disable_logging_to_file function
```

```
fx disable_logging_to_stdout
```

```
fx enable_logging_to_file
```

```
fx enable_logging_to_stdout
```

```
EXAMPLES_PATH
```

```
f
```

Upon execution, an output area will appear.

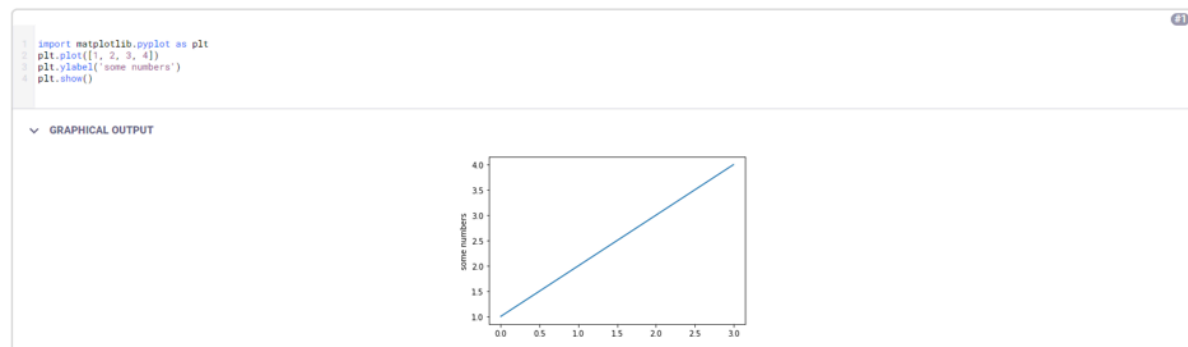
Output content can be **text**:

```
1 print("Hello World!")
```

TEXT OUTPUT

Hello World!

or **graphical**:



Additionally, an execution badge will appear on the upper right corner of a cell.

**Note** A badge will not appear on your cell until you have attempted to execute the cell at least once.

During execution, the badge will display a rotating wheel:

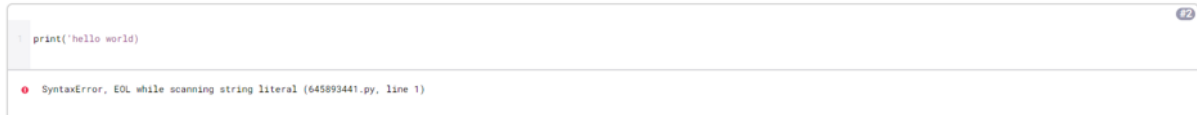


## Preview)

When execution completes, a number will appear indicating the order in which the cell was executed:

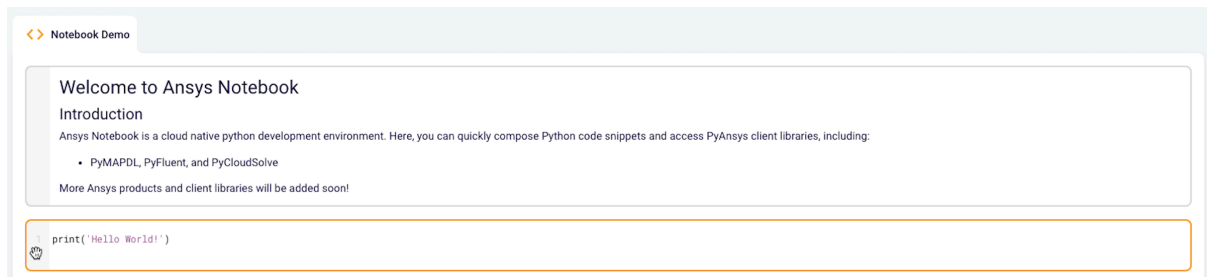


If your cell execution results in an error due to bad code, output content will not appear. Instead, the cell will report error details:

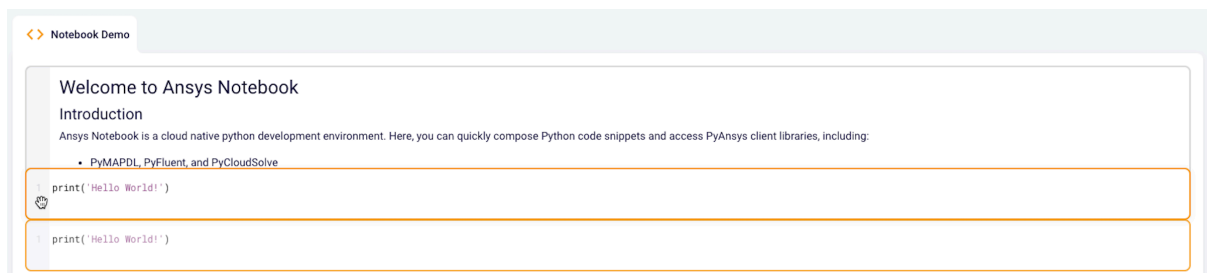


**Note** The cell badge number will still increment even if a coding error occurs during execution.

You can quickly re-order cells by drag-and-drop. Move the mouse cursor to the cell's line number area. The cursor will turn into a hand icon, confirming you can click and drag the cell:

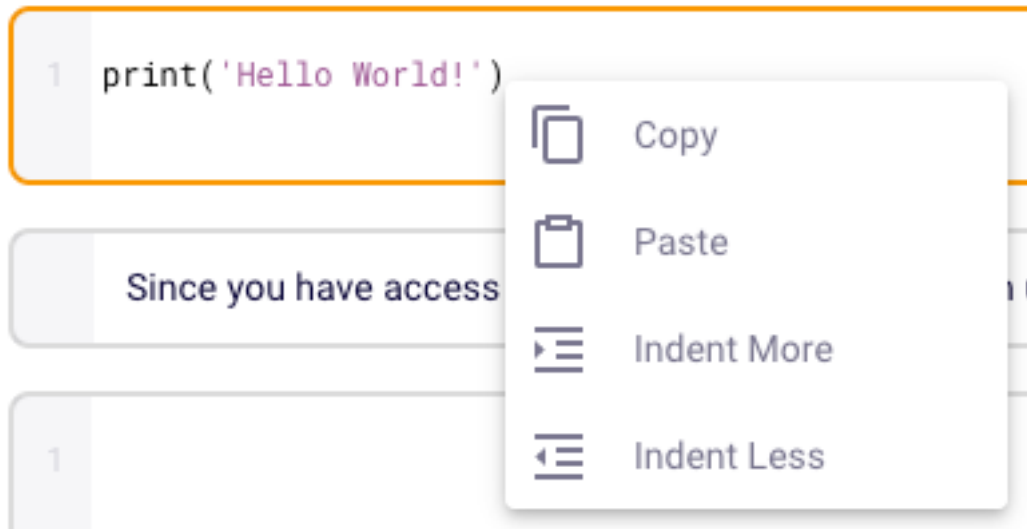


Once you click on the cell, you can drag and drop the cell above or below any other cell in your notebook.



Other Cell Actions:

By right-clicking in the content area of a cell, you can access additional cell actions through a context menu:



-  Copy

#### Copy

Adds the current cell contents to your clipboard.

-  Paste

#### Paste

Inserts your current clipboard contents into the cell.

-  Indent More

#### Indent More

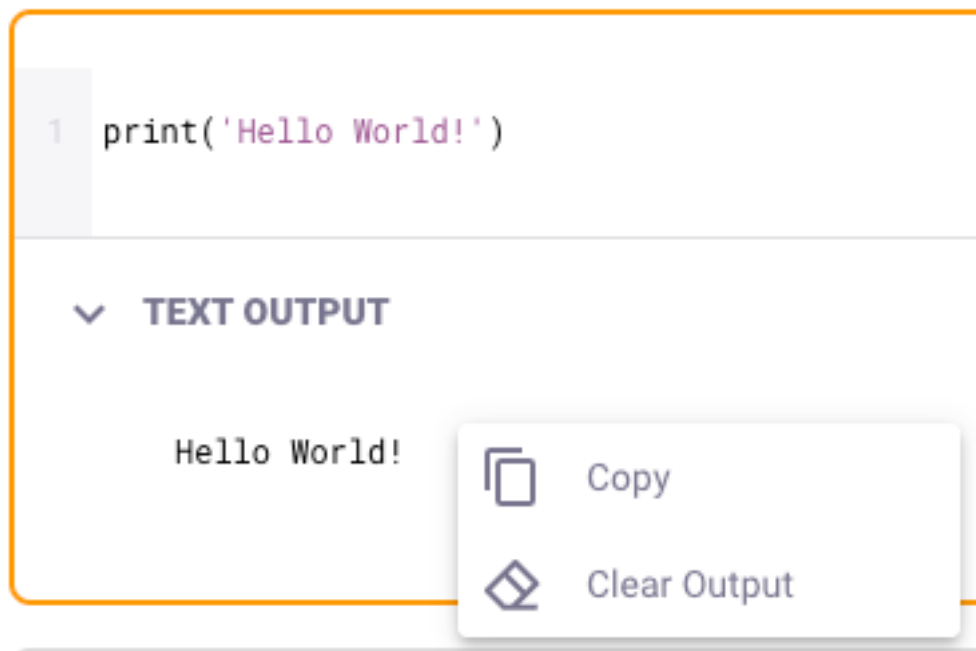
Moves all cell contents to the *right* by four spaces.

-  Indent Less

#### Indent Less

Moves all cell contents to the *left* by four spaces.

By right-clicking in the output area of a cell, you can access additional cell actions through a context menu:

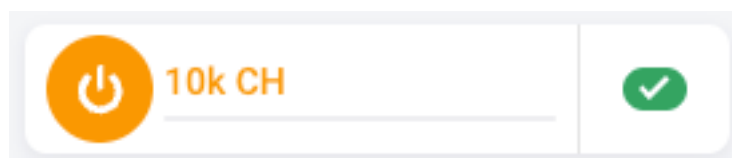


- **Copy**  
Adds the current cell output contents to your clipboard.



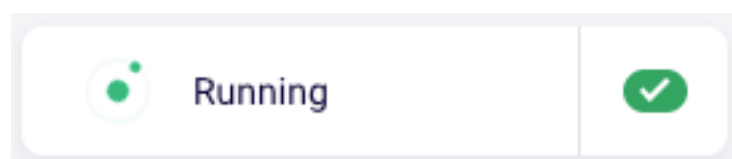
- **Clear Output**  
Clears the current cell output.

## Status



The Status area reports the current state of your notebook. By default, the Status shows the available core hours for your account. This indicates that no action is currently executing within your notebook. On the right side, you can see the current connection status of your notebook to the server.

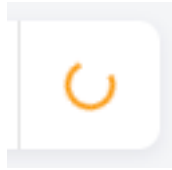
When you execute a notebook cell, the Status reports a *running* state:



The Status returns to the currently available core hours when the cell execution completes.

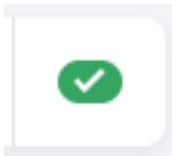
### Server Status

The Server Status shows if you are connected or not. In order to execute cells or use the Workspace, you must be connected to the server. The Server Status can be one of the following:



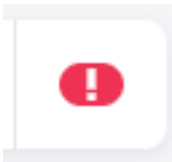
- **Pending**

Waiting to connect to the server. This status will appear when you first open a notebook.



- **Connected**

The server is up and running. You can execute cells and use the Workspace as normal.



- **Error**

The server is down. You cannot execute cells or use the Workspace. You must wait until the server is back up to access the Workspace or execute cells again.

## 1.3 Tutorials

---

**Note** Limited Preview version

---

### 1.3.1 Basic Modal Analysis Using MAPDL

Demo notebooks will be included in a shared Project at a later date.

## 1.4 Issues

---

**Note** Limited Preview version.

---

The following is a list of known issues and limitations for Ansys Notebook.

### 1.4.1 Product Instance Resources

When you launch a product instance from an Ansys Notebook cell, the cloud back end provisions a dedicated pod to run the Ansys product. The following compute resource limits apply to each pod:



Table 1.1. Pod Compute Resource Limits

Resource	Limit
CPU	8 vCPUs
RAM	32 GB requested, 64 GB max

### 1.4.2 Cell Stuck In Running State

There is a chance that a cell can get stuck in the running state indefinitely. If this happens, you can restart the kernel to reset the notebook to a clean state.

### 1.4.3 PyAEDT Initialization Delay

The first time a PyAEDT instance is launched in notebook, it may take up to ~20 minutes. Subsequent startups of PyAEDT instances will occur significantly faster.

### 1.4.4 PyAEDT Reinitialization After Deleting Instance

If you delete an AEDT instance, either from the Ansys Product Instance tab or programmatically, you will need to reset the state of the PyAEDT library before starting a new AEDT instance in the same Notebook. To reset the state, run the following code:

```
oDesktop = None
oAnsoftApplication = None
del oDesktop
del oAnsoftApplication
```

You can also reset the state by restarting the kernel.

## 1.5 Support

---

**Note** Limited Preview version.

---

Please contact Ansys Account Support.





