國立中山大學
National Sun Yat-sen University

Python Application in DIgSILENT PowerFactory

Created by Firas Quthbi Sidqi (V133010001)

Supervisor: Dr. Mohammed Manaz

Outline

- Learning reason
- Prior knowledge
- Python Preparation
- Introduction
- Tutorial_Python Chapter 1
- Tutorial_Python Chapter 2
- Python logic
- Post test

Learning Reason

(Why should we use Python in DIgSILENT PowerFactory?)

Consider you are working with this SLD

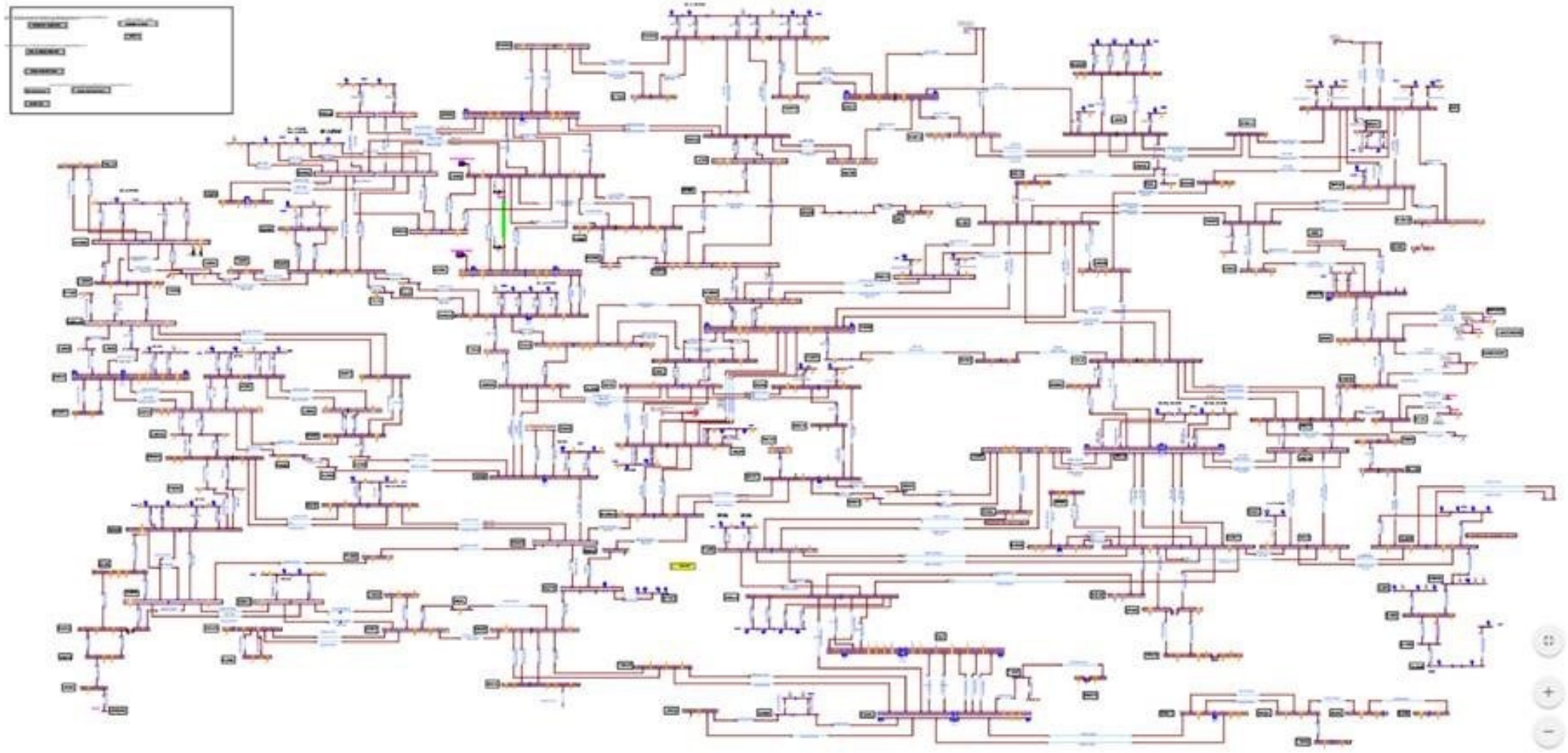


Image source: emtp

Background reason

- Suppose you want to know the best bus for one Photovoltaic (PV) placement, considering the minimum losses. Will you connect the PV one by one?
- Suppose you have 5 PV that should be installed, there will be a lot of combinations.

Prior Knowledge

(What to know before learning Python in PowerFactory)

Prior Knowledge

1. Basic modelling skills in DIgSILENT PowerFactory
2. Basic simulation skills including Load Flow, Short Circuit, RMS Simulation, Unit Commitment, etc.
3. Basic Python knowledge
4. Data structure inside DIgSILENT PowerFactory

We will not explain more about 1, 2, and 3.

We will focus more on 4 since the majority of people skip this.

Data structure inside DIgSILENT PowerFactory

Look at the Data Manager window below.

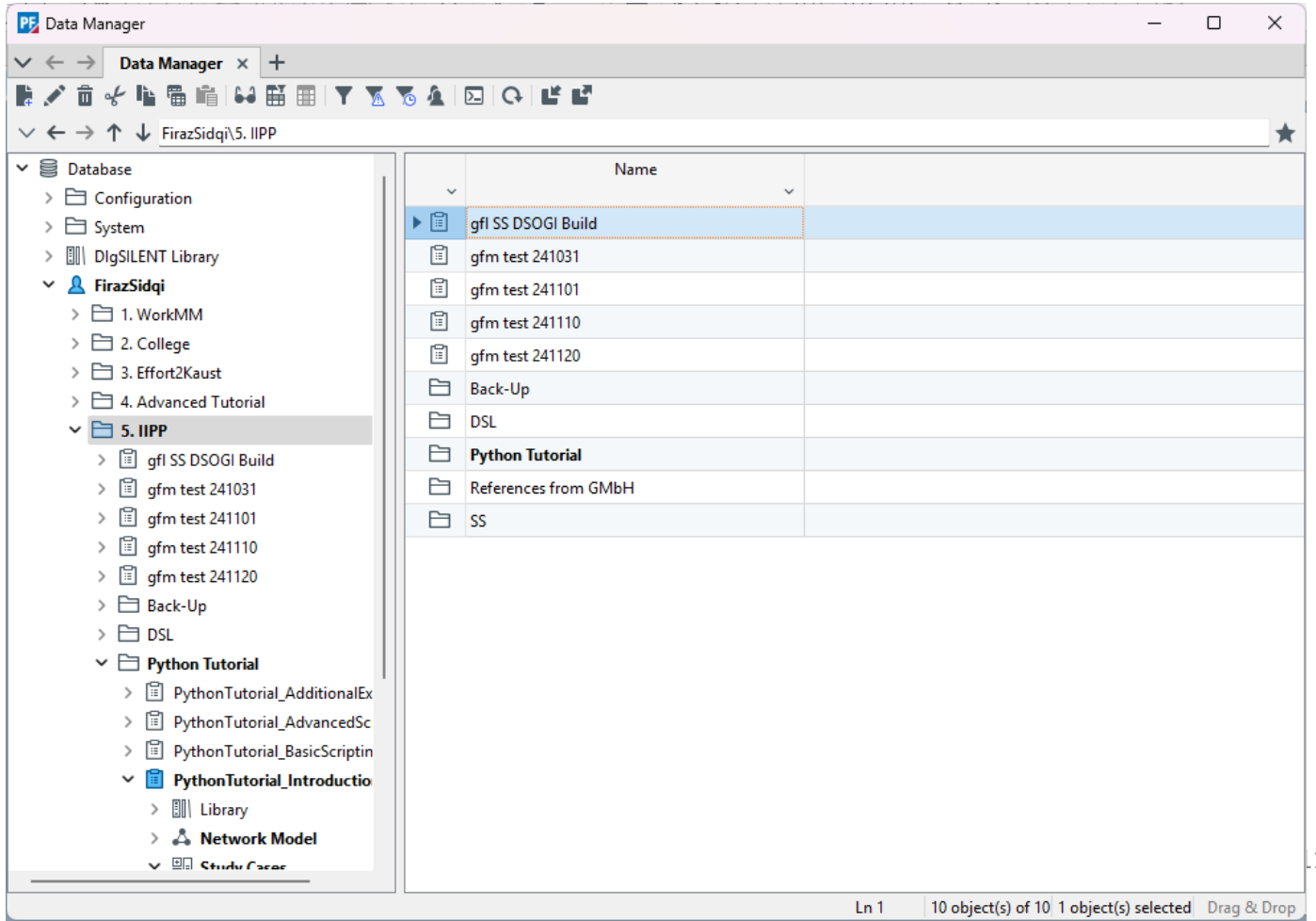


Image source: personal documentary

Compare it with the “Windows Explorer” window

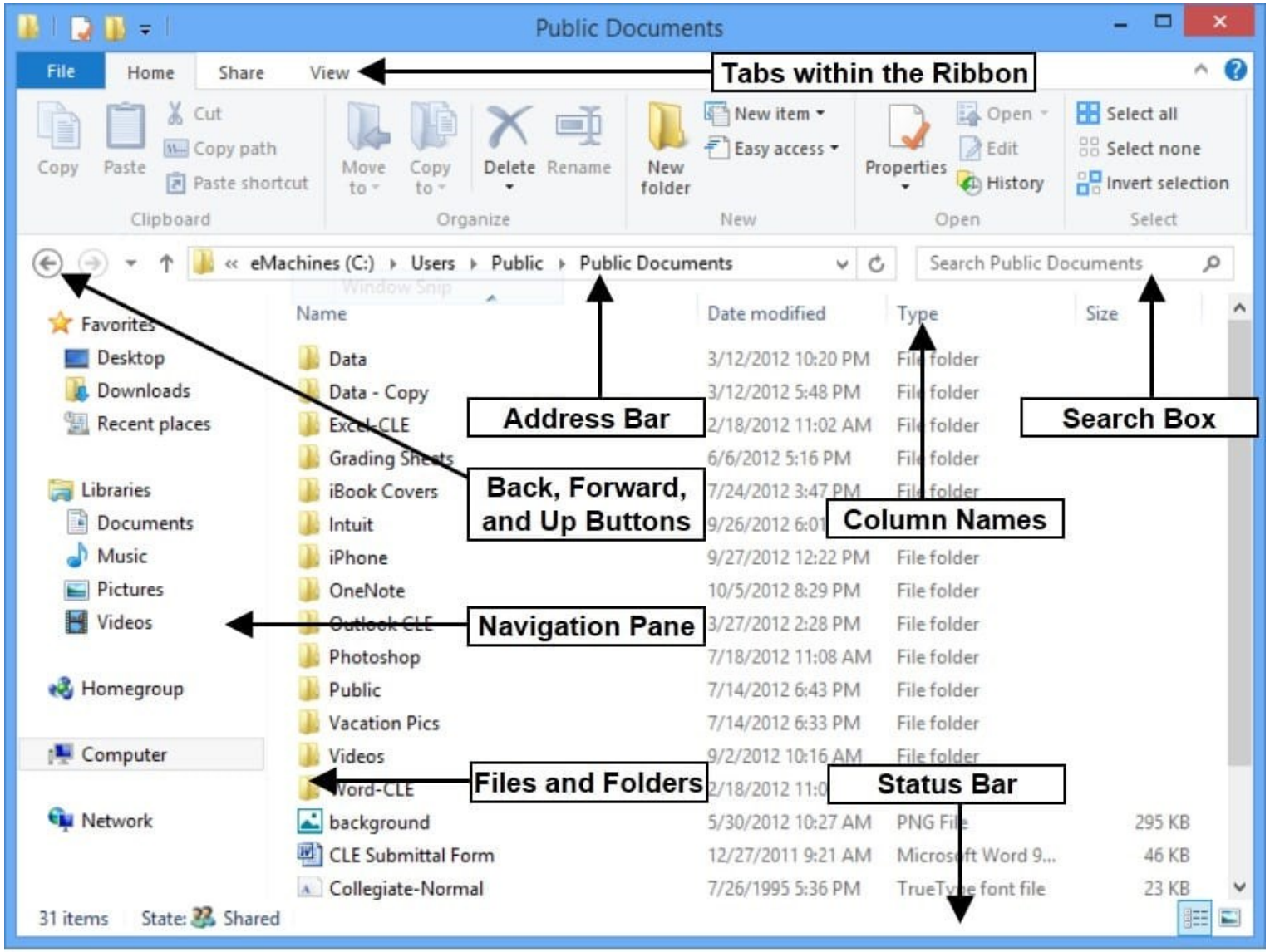


Image source: [teachucomp]

They look quite similar right?

There are several similarities between both. Such as:

- Address bar / Directory bar
- Back, forward, up and down buttons
- Navigation pane
- Files and Folders

What other similarities?

Every file in Windows Explorer has its own Extension. Such as:

- PDF files : .pdf
- Compressed files : .zip, .rar, .7z
- Microsoft Office files : .docx, .pptx, .xlsx

It is worth noting that All files inside PowerFactory Data Manager have their extensions as well. Such as:

- Project files : .IntPrj,
- Study Case : .IntCase
- Even a single component like a synchronous machine has an extension (.ElmSym)

Since you know that they are similar

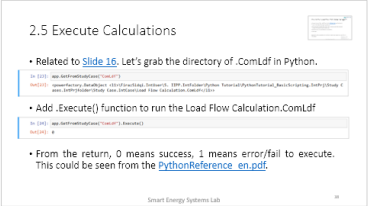
Now get used to it by starting to do these things:

- When you activate your project, use data manager.
- When you export your project, use data manager.
- When you want to delete your project, use data manager.
- When you want to navigate and organise your internal database, use data manager.
- When you want to run a load flow, use data manager.

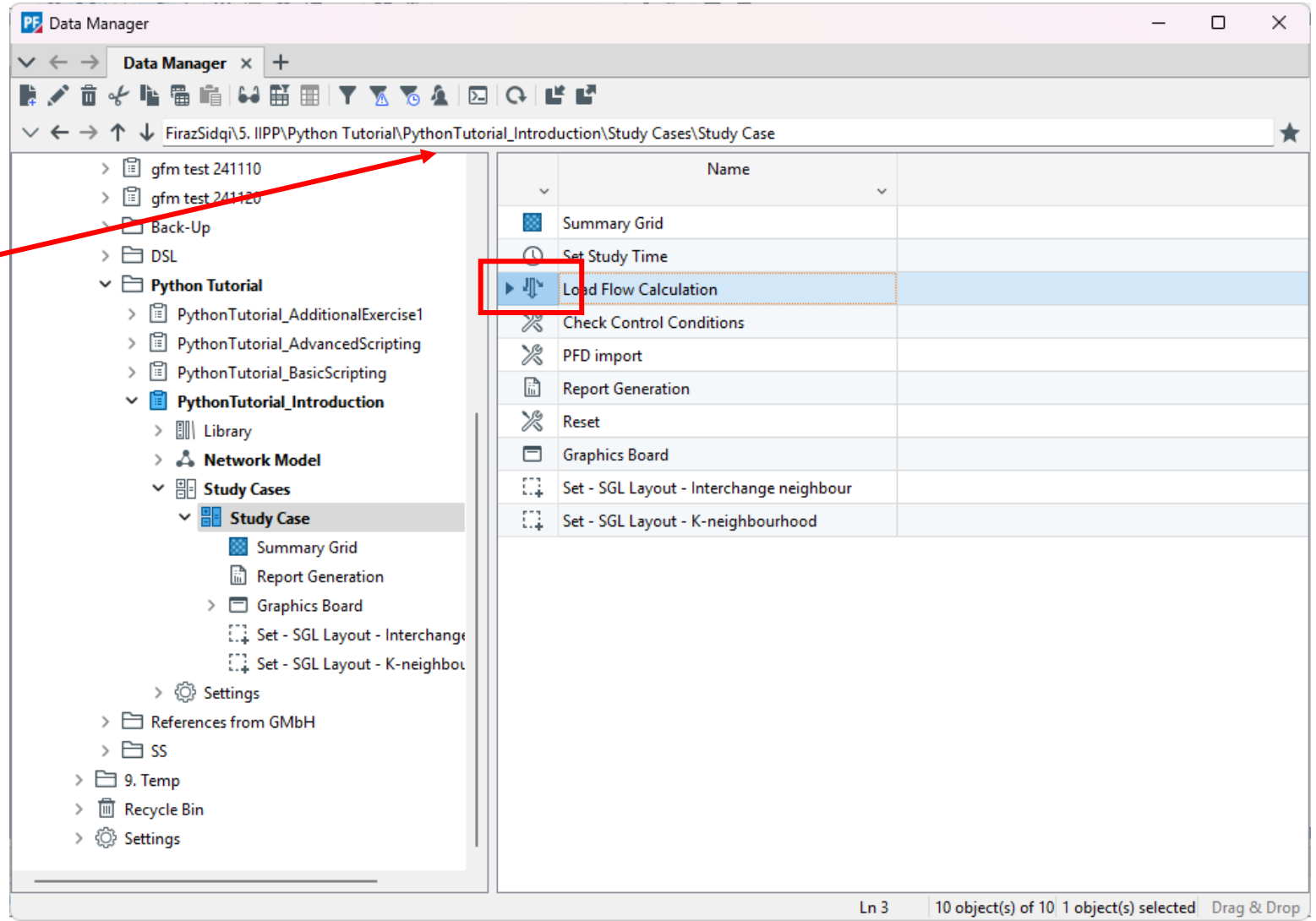
How to Run Load Flow from Data Manager?

- Import any of the .pfd files in this GitHub link and activate
- Go to the Data Manager windows
- By using the navigation pane, go to {username}\{activated project}\Study Cases\Study Case
- Double Click on Load Flow Calculation
- Execute

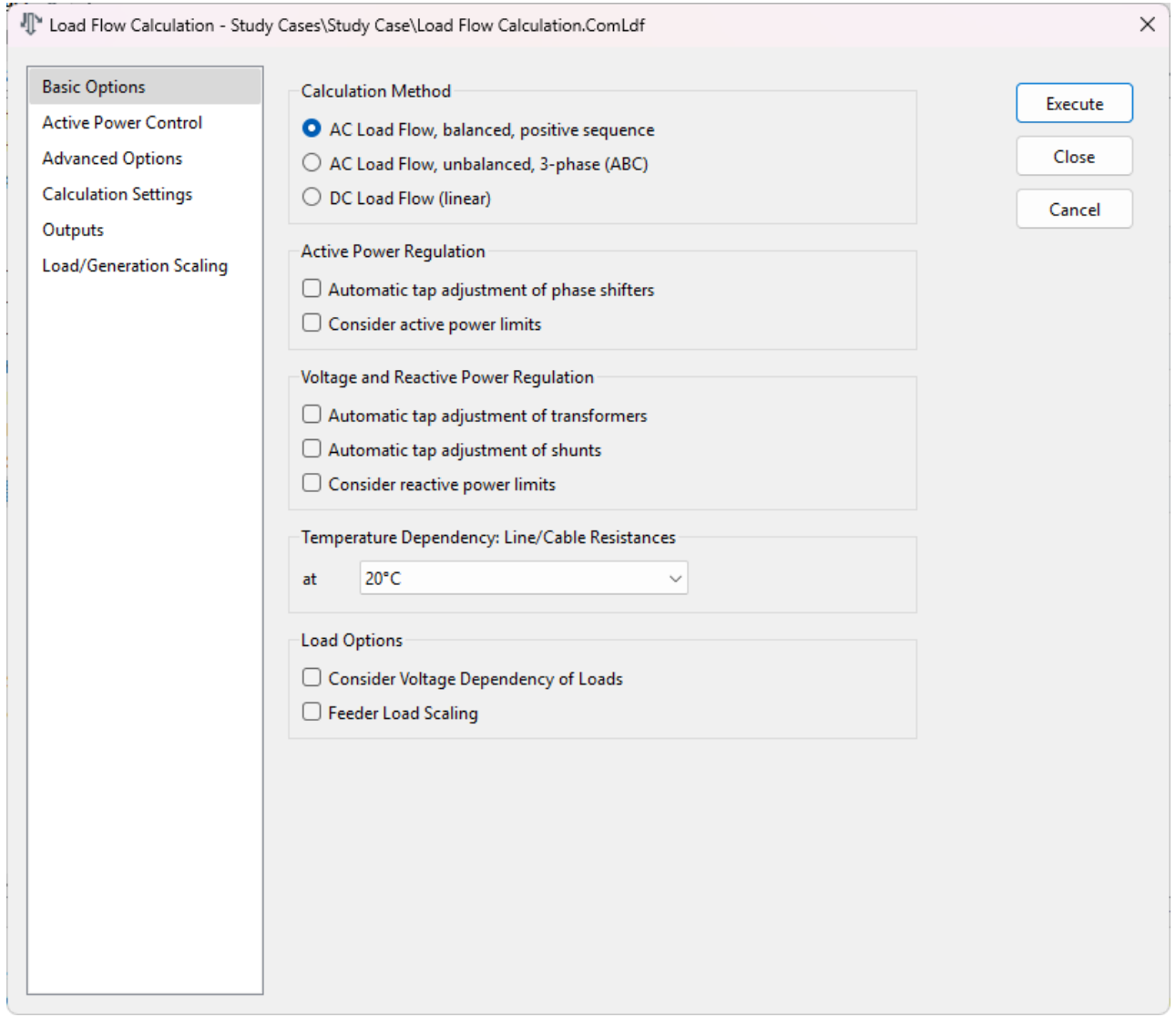
Basically, this procedure is similar to when you open an .exe file in Windows



Location of Load Flow Calculation.ComLdf

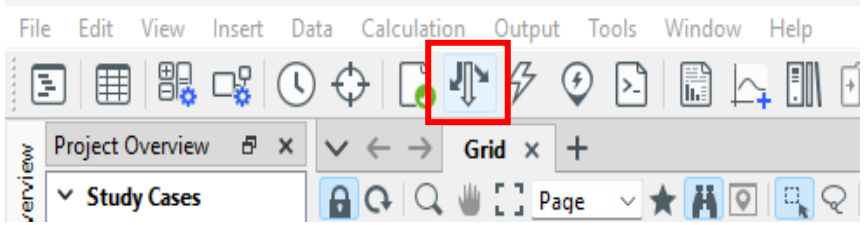


It will open the Load Flow window, Execute



So, why do it the hard way?

The quickest way to run load flow is by clicking this button, a fact that many of us are already aware of.



But in Python, you cannot click that button anymore to run the load flow. You need to open the Load Flow Calculation.ComLdf by using the location where it is stored. It is applicable not only for load flow but for everything.

We will learn about it deeper on the next section.

Python Preparation

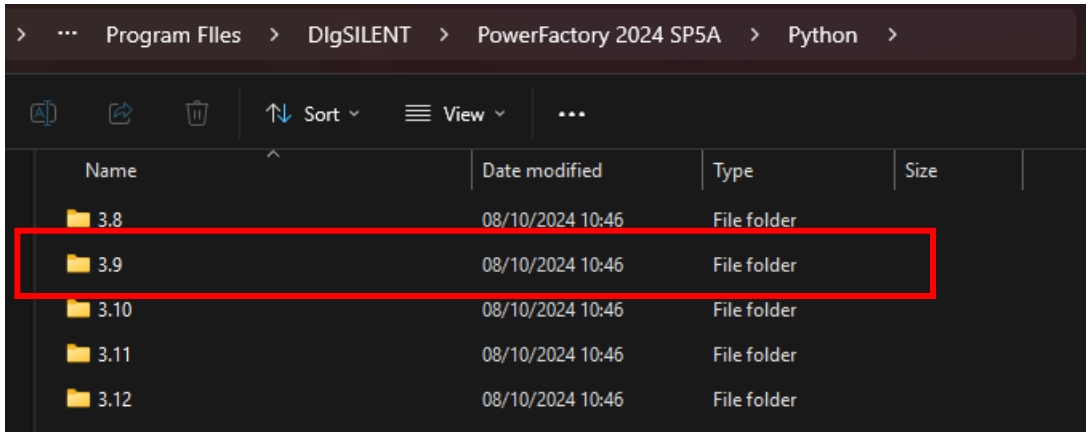
(How to install Python environment in Windows)

Set-up Python Installation

- Since DIgSILENT PowerFactory is only available in Windows, so we will set up the Python in Windows.
- Download and extract the WinPython V 3.9.10.

Why use WinPython? It is portable, simple, and complete.

Why version 3.9.10? V3.9.10 is supported on PowerFactory 2019 - 2025



The Usage of Jupyter Notebook (.ipynb)

From WinPython, open Jupyter Notebook.

There are 2 reasons to use Jupyter Notebook:

1. Cell management, useful for troubleshooting the script
2. Better IDE compared to DIgSILENT PowerFactory Script IDE

Python Introduction

(Non Interactive Mode)

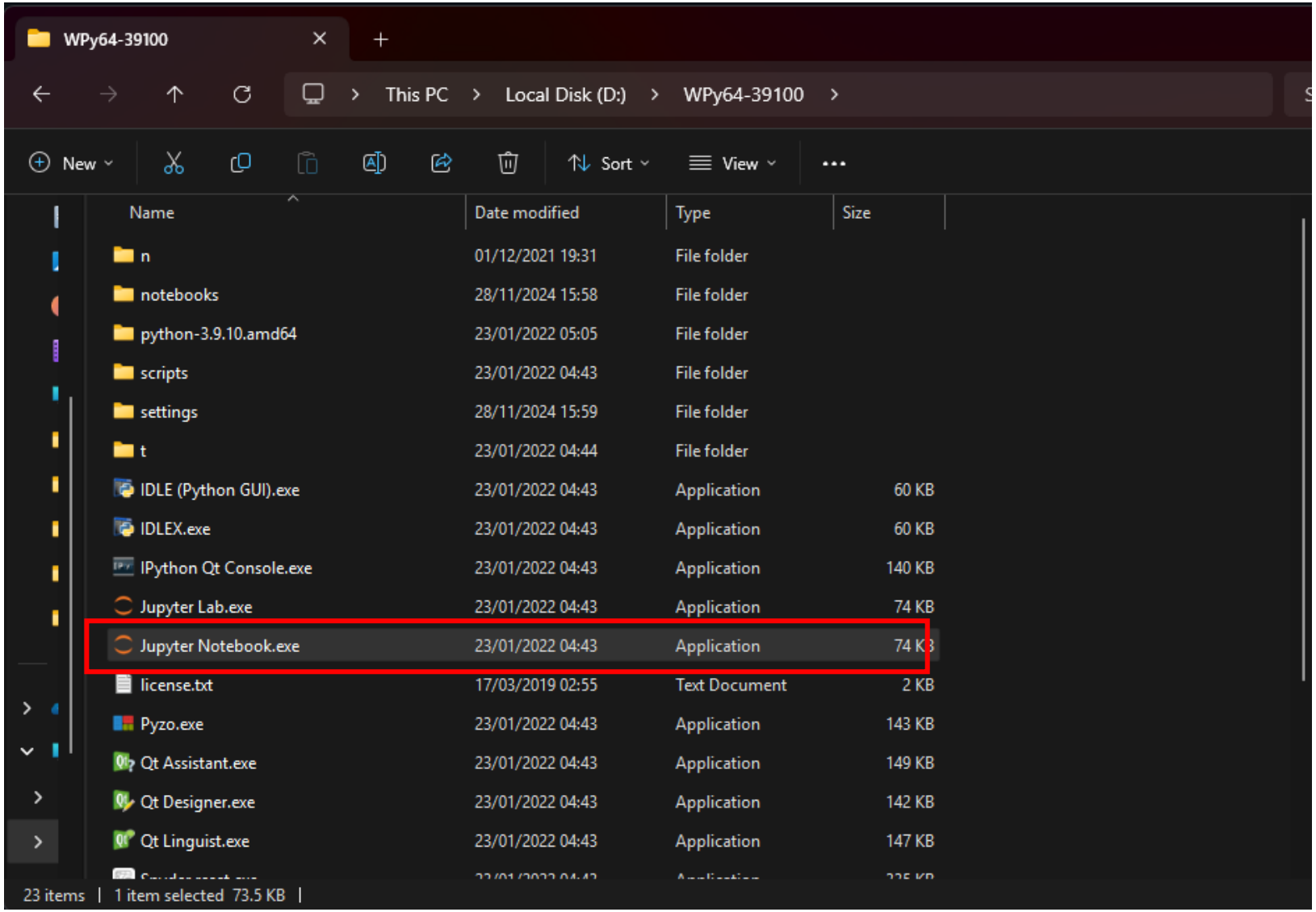
Internal Python vs External Python

In this session, we will learn how to integrate Python and PowerFactory from the external side.

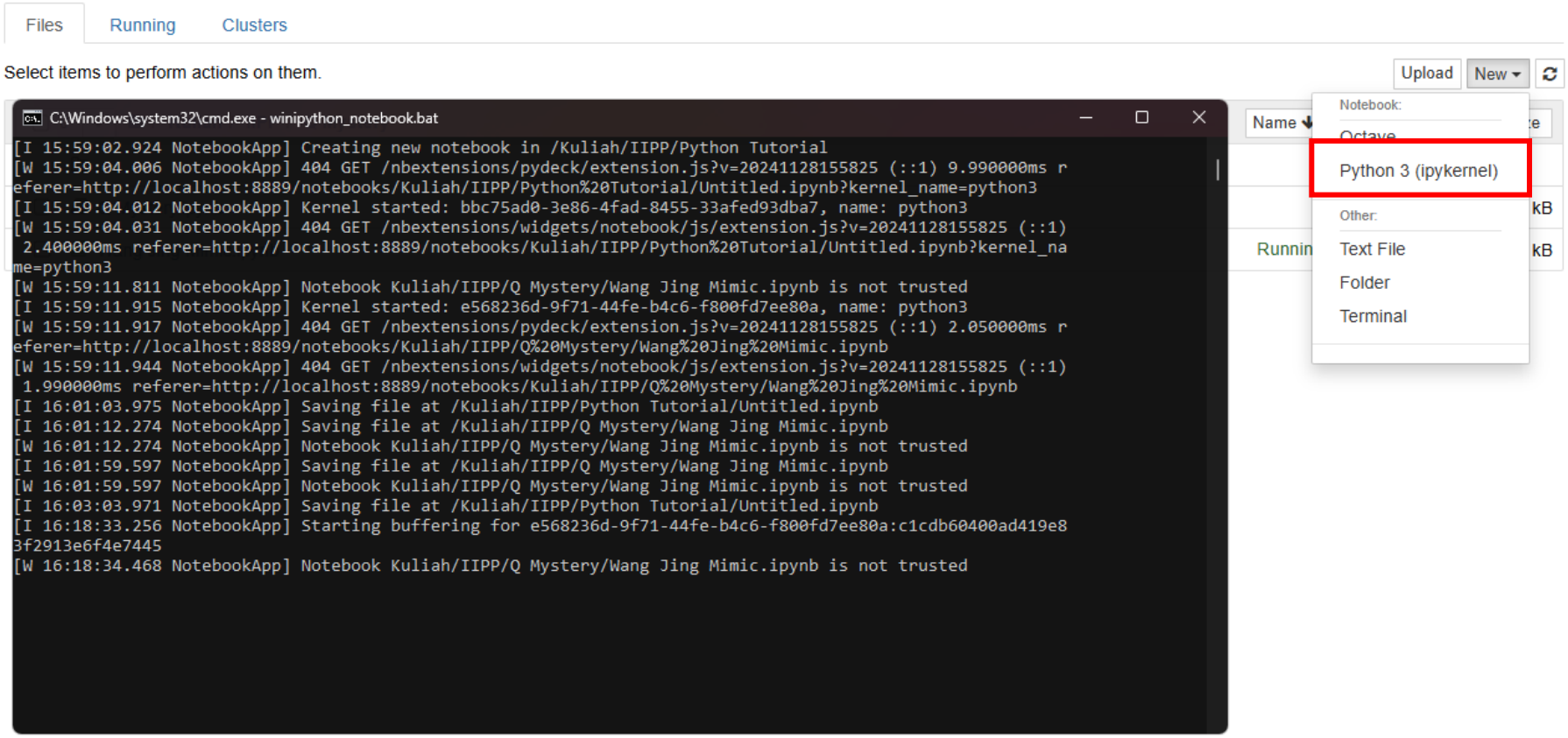
The official tutorial in Tutorial_Python.pdf utilizes an internal scripting IDE for Python. However, we believe that learning through Jupyter Notebook generally allows individuals to learn more effectively. In contrast, the internal IDE is the most suitable choice for learners of the DIgSILENT Programming Language (DPL).

So, it is needed to know that in this PowerPoint Session, we will redo everything from Tutorial_Python.pdf but in the Jupyter Notebook environment.

Open Jupyter Notebook



cmd.exe will pop up together with a browser tab. Create a new Python 3 (ipykernel)



jupyter Tutorial_Python Last Checkpoint: 21 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code Voilà nbdiff

In [2]:
```
import sys
import os
sys.path.append(r"D:\Program Files\DIgSILENT\PowerFactory 2024 SP5A\Python\3.9")
import powerfactory as pf
app=pf.GetApplicationExt()
```

In [3]:
```
app.Show()
```

- Rename the notebook to “Tutorial_Python”
- Create and type these 2 cells, change “D:\Program Files\DIgSILENT\PowerFactory 2024 SP5A\Python\3.9” with the equal directory of PowerFactory Python Module.
- Run the 2 cells, and PowerFactory non-interactive mode will appear.
- Note that the explanation of cell 1 could be obtained from [Tutorial_Python.pdf or PythonReference_en.pdf](). Below is just example of explanation.

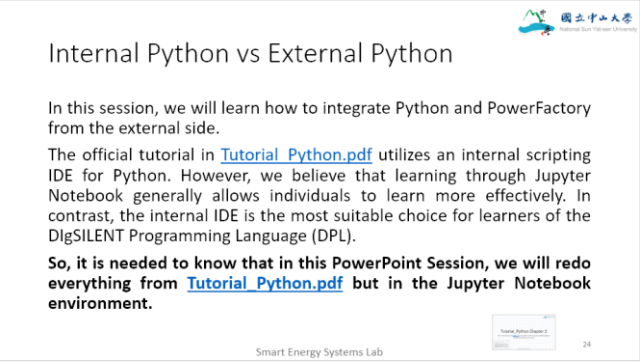
```
import powerfactory
```

The “powerfactory” module interfaces with the *PowerFactory* API (Application Programming Interface). This solution enables a Python script to have access to a comprehensive range of data available in *PowerFactory*:

- All objects
- All attributes (element data, type data, results)
- All commands (load flow calculation, etc)

Tutorial_Python Chapter 1

(As mentioned in slide 24, we will do the tutorial with Jupyter Notebook Environment)



Internal Python vs External Python

In this session, we will learn how to integrate Python and PowerFactory from the external side.

The official tutorial in Tutorial_Python.pdf utilizes an internal scripting IDE for Python. However, we believe that learning through Jupyter Notebook generally allows individuals to learn more effectively. In contrast, the internal IDE is the most suitable choice for learners of the DIgSILENT Programming Language (DPL).

So, it is needed to know that in this PowerPoint Session, we will redo everything from Tutorial_Python.pdf but in the Jupyter Notebook environment.

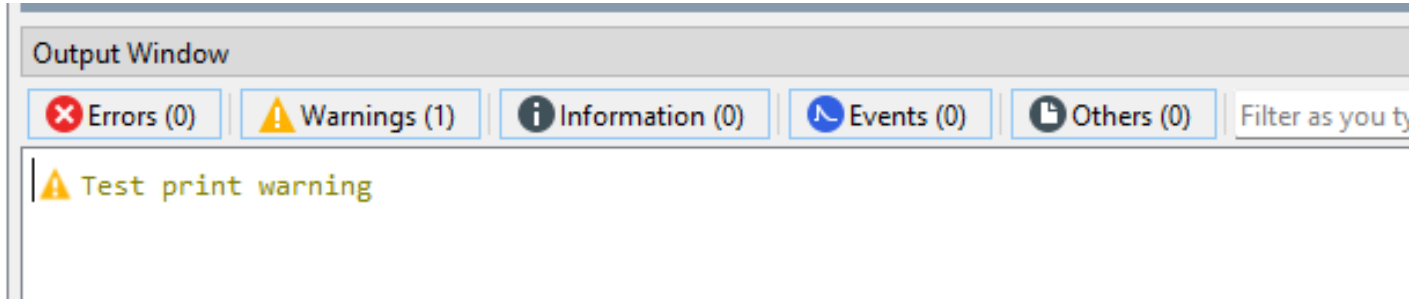
Smart Energy Systems Lab 24

1.3.2 Write messages to the output window

Let’s run Warning messages in Jupyter Notebook:

```
In [7]: app.PrintWarn("Test print warning")
```

Check in the PowerFactory output window, it should appear like this:



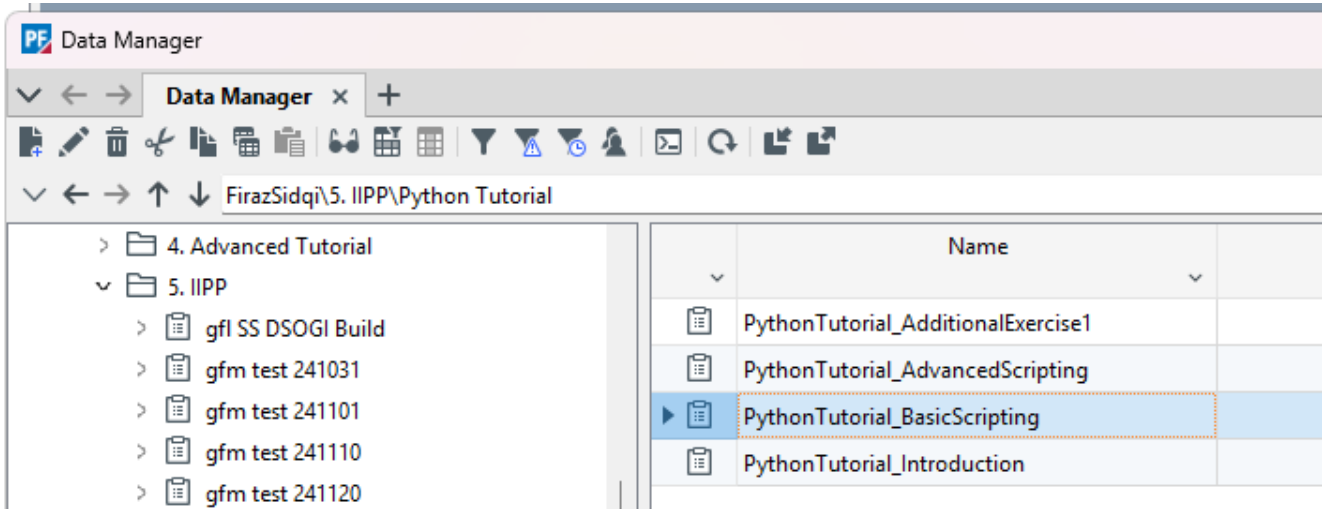
Congrats! Now your Jupyter Notebook (Python) has already integrated with PowerFactory

Tutorial_Python Chapter 2

(Basic Python Scripting)

Let’s first activate the project from Python

- Look for the project’s directory/location in the Data Manager



- For my directory, I use the script below:

In [5]:
```
app.GetCurrentUser().GetContents("5. IIPP")[0].GetContents("Python Tutorial")[0].GetContents("PythonTutorial_BasicScripting")[0].Activate()
```

Let’s first activate the project from Python

```
In [5]: app.GetCurrentUser().GetContents("5. IIPP")[0].GetContents("Python Tutorial")[0].GetContents("PythonTutorial_BasicScripting")[0].Activate()
```

Here is the main idea.

- To activate the project, we must call the project location/directory.
- app.GetCurrentUser() will call the user inside PowerFactory
- GetContents() will call the children’s folder
- If you are curious, just run app.GetCurrentUser().GetContents() and observe what is the output
- Jupyter Notebook will be the best to apply some portion of the script and try that before combining it with the bigger script

2.1 Access Network Objects

Rather than import again, directly write these lines:

```
In [15]: Lines = app.GetCalcRelevantObjects('.ElmLne') #get list of all lines
         for Line in Lines:
             print(Line.loc_name)

         Line to Load
         Line to Static Generator
         Line to Synchronous Machine
```

We do not need to import Powerfactory and define app except the Notebook is restarted.

Also instead of print the Line name in the PowerFactory output window, we can print it directly to the Notebook

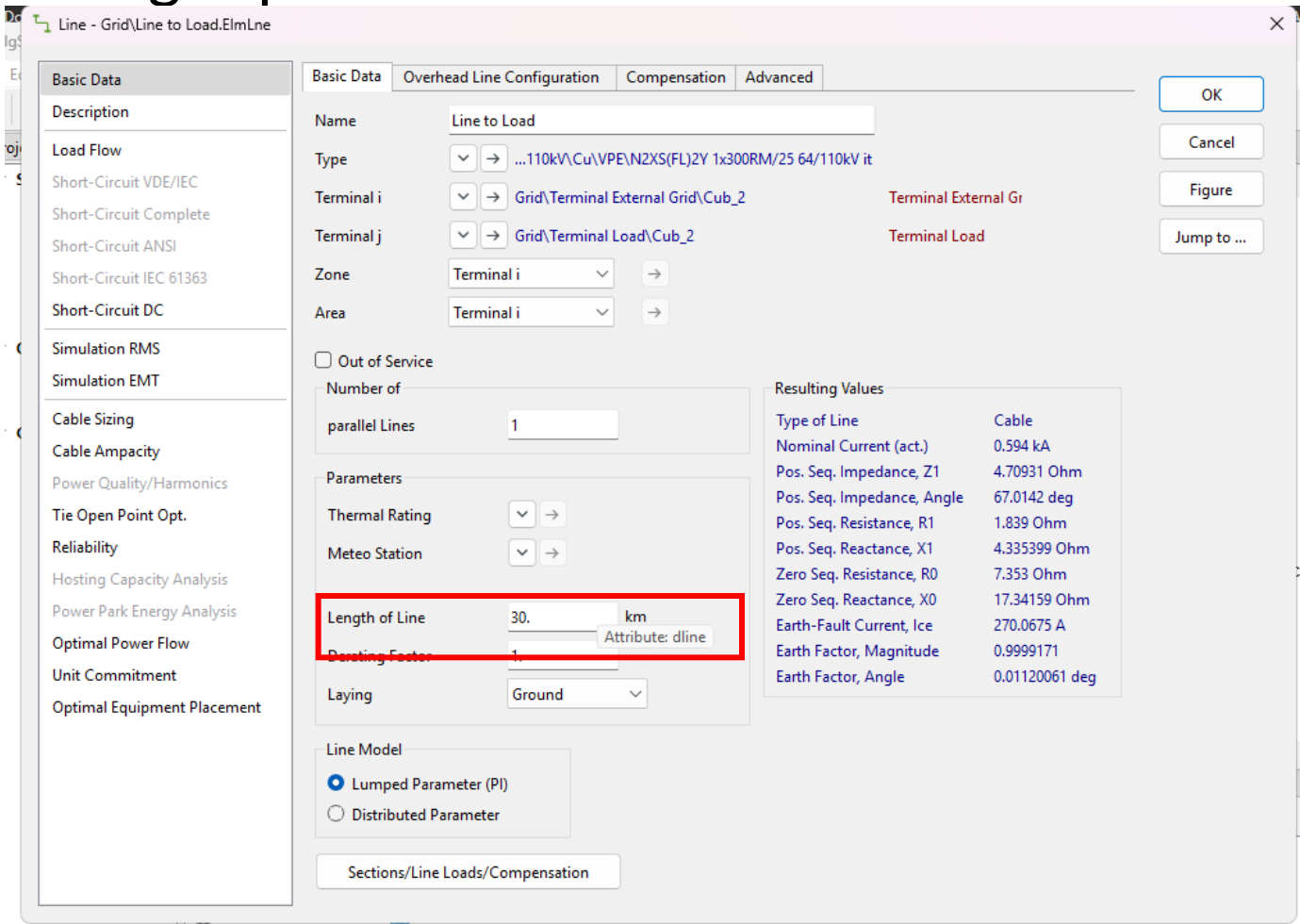
2.2 Identify, Access and Modify Object Parameters

- To identify the element parameters, we need to hover a cursor to the UI. For example, we want to change the length of “Line to Load”. We need to access the line first.

```
In [20]: Line_to_Load = app.GetCalcRelevantObjects("Line to Load.ElmLne")[0]
```

2.2 Identify, Access and Modify Object Parameters

- Then open any line, hover to the length parameter



- We got the variable dline
- Call it in the Python

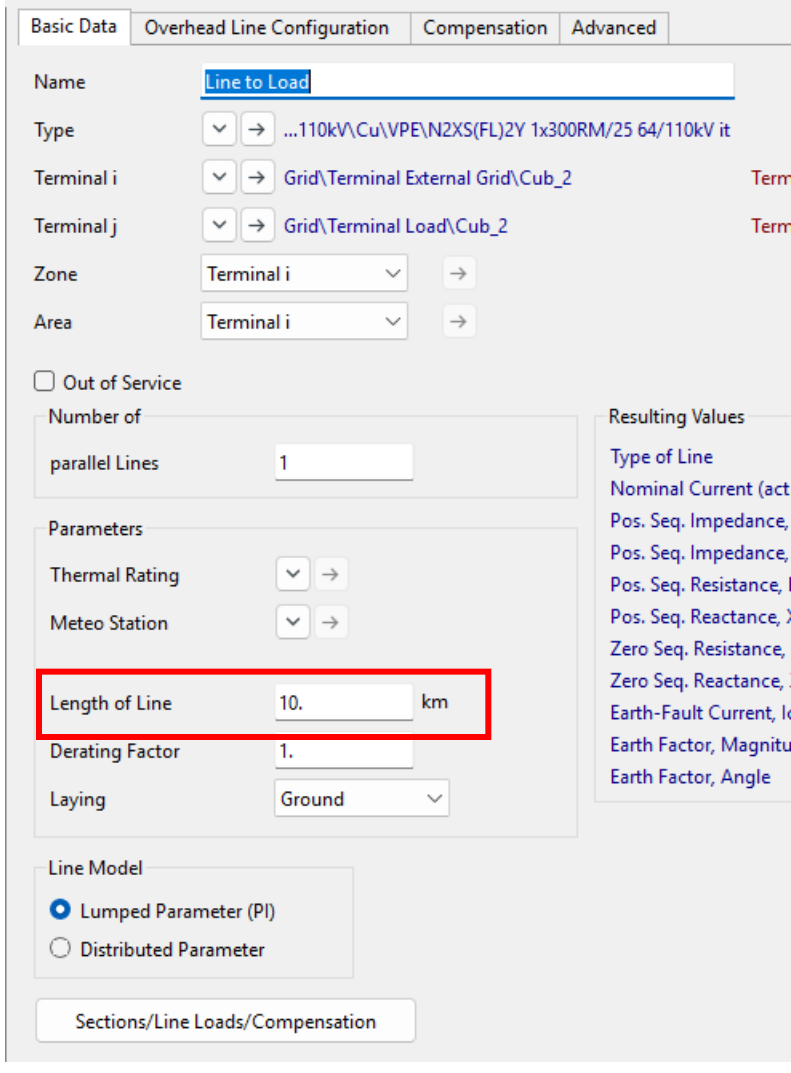
```
In [21]: Line_to_Load.dline
Out[21]: 30.0
```

2.2 Identify, Access and Modify Object Parameters

- To change the value, just do it straightforward

```
In [22]: Line_to_Load.dline = 10
```

- Check the result in the PowerFactory User Interface



2.5 Execute Calculations

- Related to Slide 16. Let's grab the directory of .ComLdf in Python.

```
In [23]: app.GetFromStudyCase("ComLdf")

Out[23]: <powerfactory.DataObject <l1>\FirazSidqi.IntUser\5. IIPP.IntFolder\Python Tutorial\PythonTutorial_BasicScripting.IntPrj\Study C
         ases.IntPrjfolder\Study Case.IntCase\Load Flow Calculation.ComLdf</l1>>
```

- Add .Execute() function to run the Load Flow Calculation.ComLdf

```
In [24]: app.GetFromStudyCase("ComLdf").Execute()

Out[24]: 0
```

- From the return, 0 means success, 1 means error/fail to execute. This could be seen from the PythonReference_en.pdf.

Python Logic

(Basic Python Scripting)

Overall Logic in Python

- All activities in PowerFactory can be done using Python, the only requirement is the directory of the object/element/variables inside PowerFactory.
- Within a few trials in this PowerPoint, I hope anyone can do the rest of the examples in the Tutorial_Python.pdf by using Jupyter Notebook IDE.

Some Tips

- Sometimes if we forget the syntax, rather than opening the PythonReference_en.pdf, pressing TAB in the middle of the syntax will give us a hint.

```
In [24]: app.Get
Out[24]: 0
```

```
GetActiveCalculationStr
GetActiveNetworkVariations
GetActiveProject
GetActiveScenario
GetActiveScenarioScheduler
GetActiveStages
GetActiveStudyCase
GetAllUsers
GetAttributeDescription
GetAttributeUnit
```

```
In [ ]:
In [ ]:
In [ ]:
```

Post Test

(Please do this after finishing the Tutorial_Python.pdf.)

Run PV Placement with minimum loss

- Download the PowerFactory project file and the Python Script in GitHub.
- Import the project.
- Move the Python Script "PLTS Placement.ipynb" to the Jupyter Notebook directory "WPy64-39100/notebooks"
- Run the Python script
- Learn from the Python script
- Create 2 or more PV systems in PowerFactory, then look for the best placement combination by targeting the minimal losses using numerical methods, AI, etc.