

/ Install with virtual environment

```
python -m venv .venv
.venv\Scripts\Activate.ps1 # Windows PowerShell.
For Linux, run source .venv/bin/activate
python -m pip install -U pip
python -m pip install ansys-lumerical-core
```

/ Starting a session

Start a Lumerical FDTD session

```
import ansys.lumerical.core as lumapi
with lumapi.FDTD() as ftdt:
    print(f"This is Lumerical FDTD {ftdt.version()}")
    input("Press any key to close session")
```

Start Lumerical Multiphysics, add CHARGE and HEAT solvers

```
import ansys.lumerical.core as lumapi
with lumapi.DEVICE() as device:
    device.addchargesolver()
    device.addheatsolver()
```

Start a Lumerical INTERCONNECT session, and print library

```
import ansys.lumerical.core as lumapi
with lumapi.INTERCONNECT() as intc:
    print(intc.library())
```

The hide flag hides the GUI during execution

```
import ansys.lumerical.core as lumapi
with lumapi.FDTD(hide=True) as ftdt:
    print(f"This is Lumerical FDTD {ftdt.version()}")
```

Run existing scripts on an existing Lumerical FDTD project

```
with lumapi.FDTD(hide=True,
    script="SetupScript.lsf",
    project="MyPyLumProject.fsp") as ftdt:
    # Use function from SetupScript.lsf
    ftdt.customsetupfunction()
    # Do analysis after running simulation
    analysisScript =
        open("AnalysisScript.lsf","r").read()
    ftdt.eval(analysisScript)
```

/ Set up simulation objects

Using keyword arguments

```
with lumapi.FDTD() as ftdt:
    ftdt.addfdtd(dimension="2D", x=0.0e-9, y=0.0e-9,
        x_span=3.0e-6, y_span=1.0e-6)
```

Using set commands

```
with lumapi.FDTD() as ftdt:
    ftdt.addrect({"name": "MyRect"})
    ftdt.setnamed("MyRect", "x", 0)
    ftdt.setnamed("MyRect", "x span", 1e-6)
```

Using OrderedDict to preserve property assignment order

```
from collections import OrderedDict
with lumapi.FDTD() as ftdt:
    props = OrderedDict([("name",
        "power"), ("override global monitor
        settings", True), ("x", 0.), ("y", 0.4e-6),
        ("monitor type", "linear x"), ("frequency
        points", 10.0)])
    ftdt.addfdftmonitor(properties = props)
```

/ Running simulations

Set up resources

```
with lumapi.FDTD(hide=True) as ftdt:
    # Add CPU/GPU resources
    cpuRes = ftdt.addresource("FDTD")
    ftdt.setresource("FDTD", cpuRes, "device type",
        "CPU")
    ftdt.setresource("FDTD", cpuRes, "name", "PyLum
        CPU")
    gpuRes = ftdt.addresource("FDTD")
    ftdt.setresource("FDTD", gpuRes, "device type",
        "GPU")
    ftdt.setresource("FDTD", gpuRes, "name", "PyLum
        GPU")
```

Run simulation locally

```
with lumapi.FDTD() as ftdt:
    # ... Create your project
    ftdt.save("MyPyLumericalProject.fsp")
    ftdt.run("FDTD", "GPU", "PyLum GPU") # Run GPU
        solver on the PyLum GPU resource
```

Run FDTD solver on [Ansys Cloud Burst Compute™](#)

```
with lumapi.FDTD() as ftdt:
    burstSettings={"account": "Example Account",
        "download": True, "Name": "MyPyLumBurstJob"}
    ftdt.run("FDTD", "GPU", "burst", burstSettings)
```

Run a MODE simulation

```
with lumapi.MODE() as mode:
    mode.addfde()
    mode.setanalysis("wavelength", 1.55e-6)
    mode.setanalysis("search", "near n")
    mode.save("MyPyLumMODEProject.lms")
    mode.findmodes()
```

Run a sweep

```
with lumapi.FDTD(hide=True) as ftdt:
    ftdt.addsweep(0) # 0 - Sweep, 1 - Optimization,
        2 - Monte Carlo, 3 - S-parmaeter Matrix, 4
        - Corner
    ftdt.setsweep("sweep", "name", "childSweep")
    ftdt.setsweep("childSweep", "type", "Ranges")
    ftdt.setsweep("childSweep", "number of points",
        10)
    # Add parameters to existing object named film
    sweepParams={"Name": "thickness", "Parameter":
        "::model::film::z max", "Type": "Length",
        "Start": 0.05e-6, "Stop": 0.15e-6}
    ftdt.addsweepparameter("childSweep", sweepParams)
    # insertsweep adds as a parent for sweeps, child
        for other types
    ftdt.insertsweep("childSweep")
    ftdt.setsweep("sweep", "name", "parentSweep")
    # Add result to existing monitor named monitor
    sweepResult1={"Name": "T", "Result":
        "::model::monitor::T"}
    ftdt.addsweepresult("parentSweep", sweepResult1)
    ftdt.runsweep("parentSweep")
```

/ Visualize results

Visualize monitor data using matplotlib Python library

```
import numpy as np
import matplotlib.pyplot as plt
with lumapi.FDTD(hide=True) as ftdt:
    # Retrieve result from Lumerical dataset
    eFieldRes = ftdt.getresult("monitor", "E")
    x, y = eFieldRes['x'], eFieldRes['y'] #
        Convert to um
    Ex, Ey, Ez = eFieldRes['E'][:, :, 0, 0, 0],
        eFieldRes['E'][:, :, 0, 0, 1],
        eFieldRes['E'][:, :, 0, 0, 2]
    # ftdt.getelectric() also gets amplitude
    eFieldAmplitude = Ex ** 2 + Ey ** 2 + Ez ** 2
    X, Y = np.meshgrid(x, y) # Create meshgrid
    plt.contourf(X, Y, eFieldAmplitude)
```