📝 Lesson Modules

The table below is the initial planning for teaching materials. May be subject to change if notice some students are lagging or upon student request for a more in-depth look. The main goal is **not to force students to learn and master these topics**, rather to *expose them to these libraries with technical-depth*, while also providing Jupyter notes and exercises.

Week	Content
1.	✅ **How Python Works, Anaconda, Conda Environments (Online Recording)** ☐ How Python Works ○ Explain low-level how Python works (interpreted language) ○ Written Code ⟹ Compiler ⟹ Bytecode ⟹ PVM (Intepreter) ⟹ CPU ○ Explain why slow but versatile (interpreted, dynamic, garbage collection) ○ Modern day use case ☐ Command line extreme basics ☐ Python environment ○ What are library and packages (where it saved) ○ Environment Creation (Pip, Venv) ○ Pip install, uninstall, update ○ Explain importance package management ☐ Anaconda & Conda ○ Explain what is Conda, and benefits (Conda VS Pip) ○ Live-demonstration of installation ○ Basic commands. ○ Creating a new conda environment. ○ Changing fetch repository (conda-forge). ○ Creating environments using YML files.
2.	✅ **Jupyter Notebook, VS Code (Online Recording)** ☐ Ipython ○ Short history, benefits ○ Usage in command line ☐ Jupyter Notebook ○ Successor to Ipython ○ Markdown, equations, link and embed ○ Kernel, Python, Instances ○ Magic commands, Terminal ○ Keyboard-shortcuts (power user) ☐ VS Code ○ Explain text-editor ○ Live-installation

	○ Basic configuration ○ Useful extensions ○ Jupyter in VS Code
3.	✅ **Git & Github** □ Git ○ History ○ Live-installation ○ Config, Branch, Add, Commits, Logs, Diff, Restore, Reset ○ Conda & Alias on Git Terminal (Windows compatible) □ Github ○ Ask them to make account beforehand ○ Creating a repository ○ Burner email ○ Linking local with remote repository (Set-Upstream) ○ Push, Fetch, Merge, Clone
4.	✅ **Technical Python** □ File Handling (CWD, File Paths, Read/Write, Binary Files vs Text Files, Storing basic data with raw binary and JSON) □ Modern Language Features (f-strings, generators, list comprehension, pattern matching, match statement, lambdas) □ Dynamic arguments (*args, **kwargs) □ Multiple file projects □ functools, itertools □ Short note on type hints and type-safe Python □ Short note on reading error messages
5.	✅ **Numpy** □ Array (Contiguous memory, Copy) □ Operation (Addition, Multiply, Matmul, Dot, Infer, Max, Min, All, Any) □ Array Manipulation (Transpose, Stack, Vstack, Column Stack) □ Boolean Mask (Comparison Operator, Where) □ Slice & Indexing □ Random (Modern version, Distribution, Seed) □ Statistical Method (Mean, Median, Var, Stdev, Covariance) □ Linear Algebra (Inverse, Eigenvalue, Norm) □ Caching
6.	**Assessment Numpy (Physics/Maths)** □ Take home questions for numpy.
7.	✅ **Matplotlib** □ Backend □ Figures Cartesian-Polar, Backend & Axes (in-depth) □ Explain Pyplot, Artist, Patches

	□ Plot & Error-Bars, Histogram, Contour, Sidebars □ Labels (Title, Legend, Gridlines) □ Fonts (Latex & Normal) □ V-Lines, Limits & Ticks □ Shading □ Annotations (Arrow, Text) □ 1 Slider
8.	✅ **Plotly & Matplotlib Animate** □ Plotly 3-D interactive graph (Basic 3D, no surface) □ Process ○ Get Data for Output ○ Define Update Function ○ Call Animation □ Animate Lines and Scatter □ Rotate angle □ Frame theory □ Animate contour & 3D (data given)
9.	✅ **Scipy** □ Constants (Reduced Planck, Atm, Torr, …) □ Special Function (Bessel, Legendre, Gamma, Beta) □ Numerical Integration □ ODE, System of ODE □ Curve fitting, Covariance
10.	✅ **Data I/O** □ pandas: DataFrame and tables □ Table manipulation □ Saving and loading data □ pandas integration with numpy and matplotlib □ pillow: Image data I/O □ pillow integration with numpy; view image with matplotlib □ Basic image manipulation
11.	**Simulation Project (Simple)** □ Formulate an equation for the system, solve it, plot and animate (e.g. projectile motion,wave equation, diffusion eq, etc…)