Python Workflows

For anyone who has Python installed and is deciding how to work in it. Assumes you've been through Setup & Installation.

You'll work with two kinds of Python files: **notebooks** for analysis and **scripts** for data acquisition and instrument control. Knowing when to use each matters more than which editor you open them in.

1 Notebooks vs. Scripts

1.1 Notebooks (.ipynb)

Notebooks combine code, output, and documentation in one file. You write code in cells and run them one at a time, seeing results immediately. Reach for a notebook for:

- Learning a new concept or experimenting
- Exploratory data analysis
- Curve fitting and making plots
- Documented analysis workflows
- Figures for lab reports

1.2 Scripts (.py)

Scripts are plain-text files that run from start to finish. Reach for a script for:

- Real-time data acquisition
- Automated measurements
- Long-running experiments
- Code that controls hardware

1.3 Why Scripts for Data Acquisition?

For acquisition and instrument control, scripts beat notebooks because:

1. **Reliability** — they run without the overhead and hidden state of a notebook kernel.
2. **Error handling** — easier to guarantee cleanup (closing instruments, files) if something fails.
3. **Automation** — they run from the command line, unattended.
4. **Version control** — plain text diffs cleanly in Git.

A useful rule of thumb: **explore in a notebook, automate in a script.** Once a process is settled enough to run end-to-end without you confirming each step, it belongs in a script.

1.4 Quick Reference

Task	Use
Learning a new concept	Notebook
Quick data exploration	Notebook

Task	Use
Fitting and plotting data	Notebook
Lab-report figures	Notebook
Real-time data acquisition	Script
Automated measurements	Script
Long-running experiments	Script
Controlling hardware	Script

2 Recommended Tools

2.1 VS Code (recommended)

VS Code handles both notebooks and scripts, which is why it's our recommended environment. Setup & Installation covers installing it and the three extensions you need.

- **Run a notebook:** open any `.ipynb` file and run cells with `Shift+Enter`.
- **Run a script:** open a `.py` file and click the play button (top right) or press `F5`.

2.1.a Useful shortcuts

- **Run cell:** `Shift+Enter`
- **Select Python interpreter:** click the Python version in the bottom status bar
- **Open terminal:** `Ctrl+`` `
- **Command palette:** `Ctrl+Shift+P`
- **Go to definition:** `F12` or `Ctrl+Click`

2.2 JupyterLab (alternative)

A browser-based environment for notebooks — a good fit if you prefer the classic notebook experience.

```
py -m jupyter lab
```

Useful shortcuts, most of which work in VS Code notebooks too. *Command mode* is when the cell border is blue — press `Esc` to get there, `Enter` to go back to editing:

- **Run cell:** `Shift+Enter` (and move on) or `Ctrl+Enter` (and stay)
- **Add a cell above / below:** `A` / `B` in command mode
- **Delete a cell:** `D D` in command mode
- **Switch to Markdown / back to code:** `M` / `Y` in command mode
- **Undo deleting a cell:** `Z` in command mode
- **Interrupt a running cell:** `I I`; **restart the kernel:** `0 0`

> **Note**
>
> JupyterLab only runs notebooks. For scripts, use VS Code.

2.3 Google Colab (alternative)

Google Colab runs notebooks in your browser with nothing to install — handy when you're on a machine without Python.

> **Limitation**
>
> Colab runs on Google's servers, so it **cannot talk to lab hardware**. Use it for analysis only: collect data on a lab computer, then upload your data files to Colab.

3 File Organization

A little structure pays off quickly. Keep each project's data, analysis notebooks, and acquisition scripts together:

```
my-lab-work/
├── experiment-1/
│   ├── data/              # raw data files (CSV)
│   ├── acquire.py         # data-acquisition script
│   ├── analysis.ipynb     # analysis notebook
│   └── figures/           # saved plots
├── experiment-2/
│   ├── data/
│   ├── analysis.ipynb
│   └── ...
└── ...
```

Back to Python for Lab Work