



Higher School of Technology

Installation

Why Python?

Python has become the go-to language for financial data analysis, machine learning, and AI—and here's why:

1. Simple, Readable, and clean syntax

- That feels like writing English
- Faster learning for non-programmers

2. All-in-One for Financial Analytics data lifecycle:

- Loading, Cleaning and preparation data using pandas, numpy
- Modeling & Prediction data using scikit-learn, prophet, keras
- Visualization of data using matplotlib and seaborn

3. Massive Ecosystem & Community

- 350,000+ open-source packages
- Global support from finance professionals, academics, and developers
- Tons of free resources: tutorials, GitHub projects, forums
- If you need it, someone's already built it.

4. Widely Used in the Finance Industry

- Python is used by JPMorgan, BlackRock, Citadel, Goldman Sachs, and top fintech companies
- Powers quantitative models, AI-based credit scoring, fraud detection, and more...
- Open-source = no vendor lock-in

Why WinPython

As an open-source language, Python supports a variety of IDEs and editors—ranging from simple to highly advanced. We choose to use WinPython (often referred to as WinPy) framework which is a portable Python distribution specifically designed for

Windows. It's particularly useful for data science tasks. Here are the main benefits of using WinPython:

1. Portable and Standalone

- No installation required: You need just to extract and then run it from any folder without any installation or configuration.
- Rapid implementation ideal for classrooms, workshops.

2. Pre-packaged Scientific Libraries

- Comes with key libraries like NumPy, SciPy, Pandas, Matplotlib, SymPy, Scikit-learn, Jupyter, and more.
- Comes with WinPython Control Panel and tools like pip for easy package management.
- Saves time by avoiding manual setup for scientific or data-heavy workflows.

3. Integrated IDEs


- Includes Spyder and JupyterLab Notebook.
- Optionally integrates with VS Code, or you can use your own editor.

5. Offline Friendly

- Since everything is pre-packaged, it's perfect for offline use or environments with limited internet access.

Download and Installation

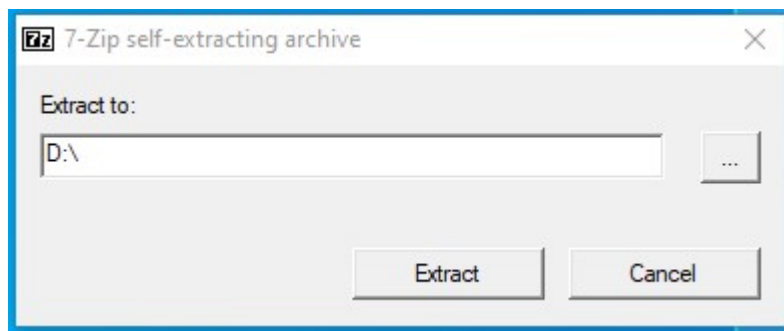
Visit winpython.github.io website and download via SourceForge the last version of WinPython distribution for your operating system (Windows/Mac OS/Linux).



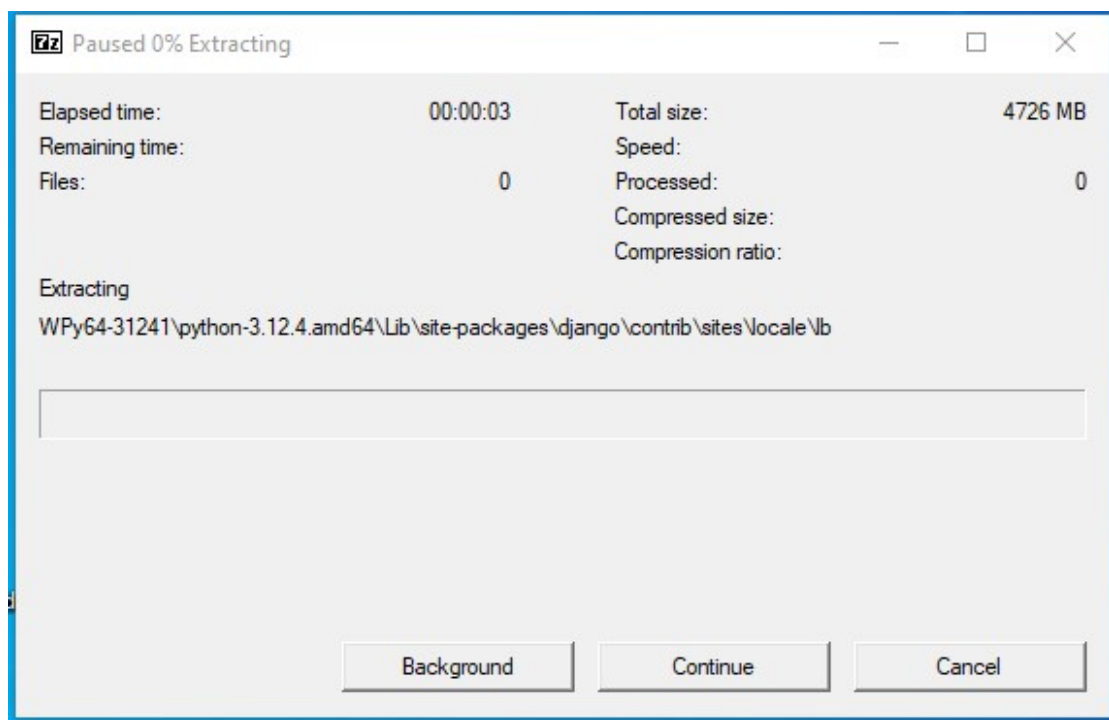
The screenshot shows the WinPython website. At the top, there are navigation links: "releases", "overview", and "portable". The WinPython logo is prominently displayed. Below the logo, a tagline reads: "The easiest way to run Python, Spyder with SciPy and friends out of the box on any Windows PC, without installing anything!". The main content area is titled "Recent Releases" and includes the following information:

- Project Home is on [Github](#), downloads pages are on [Sourceforge](#) and [Github](#), [md5-sha](#), [Discussion Group](#)
- Release [2025-01](#) of March 1st, 2025
- Highlights (*): Python-3.12.9, Python-3.13.2, [scipy-1.15.1](#), numba-0.61.0, jupyterlab-4.3.5, scikit-learn-1.6.1
- WinPython 3.12 Downloads (**) via [SourceForge](#) and [Github](#)
- WinPython64-3.12.9.0dot = Python 3.12.9 64bit only : [Changelog](#), [Packages](#)
- WinPython64-3.12.9.0slim = Python 3.12.9 64bit with PyQt5 + Spyder - Torch: [Changelog](#), [Packages](#)
- WinPython 3.13 Downloads (**) via [SourceForge](#) and [Github](#)
- WinPython64-3.13.2.0 = Python 3.13.2 64bit only : [Changelog](#), [Packages](#)
- WinPython64-3.13.2.0slim = Python 3.13.2 64bit only : [Changelog](#), [Packages](#)

As noted earlier, installing WinPython is easy, just run the executable file, and click extract:



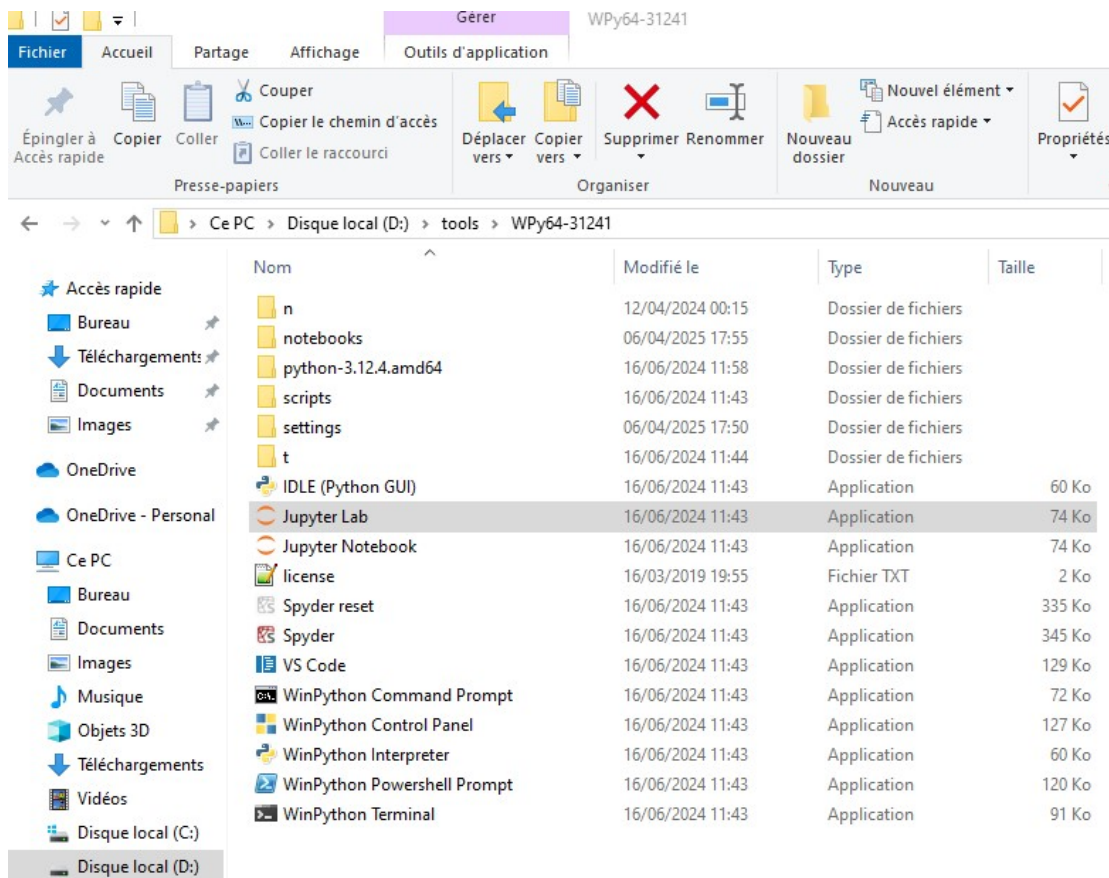
Extraction takes some time...



Using Jupyter

JupyterLab is a software that runs in your browser and allows you to do a variety of things such as: edit text, view files, and (most importantly) work with Jupyter notebooks.

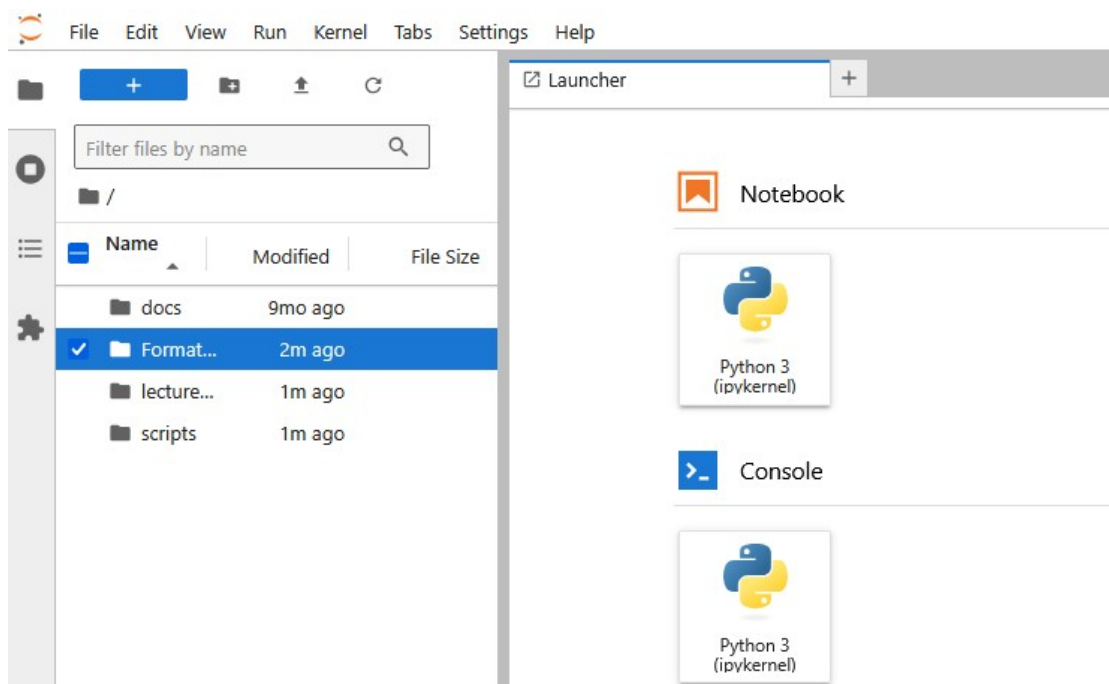
Browse to the WinPython installation directory and start JupyterLab.



Opening a Jupyter Notebook

Once the web browser is open, you should see the JupyterLab dashboard. This page shows the file system of the machine running the Jupyter server and allows you to navigate and open particular Jupyter Notebooks (or other types of files!).

The dashboard page is shown below.



Install packages

In addition to WinPython, the Python distribution comes with package management tool `pip` that help you to have the right packages and keep them all updated.

We will work through an example below how to install a new package needed for some later labs. Generally, packages can be installed by using :

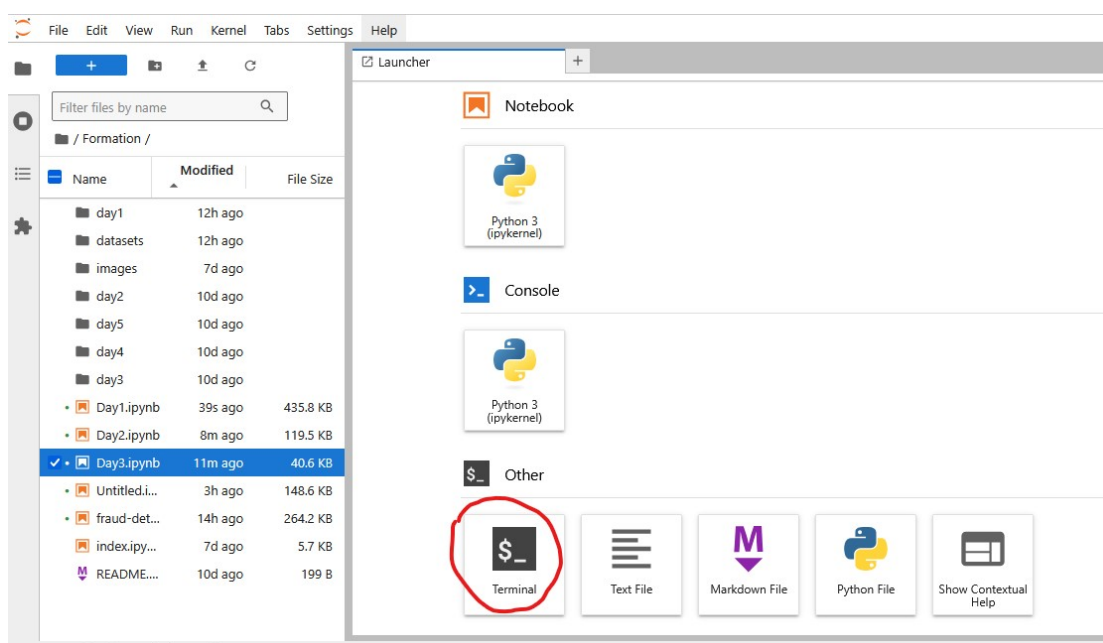
```
pip install <package name>
```

You can also update a package by running :

```
pip install <package name> --upgrade
```

Please install the first package you will need later by following the instructions below:

- Open a WinPython Terminal.



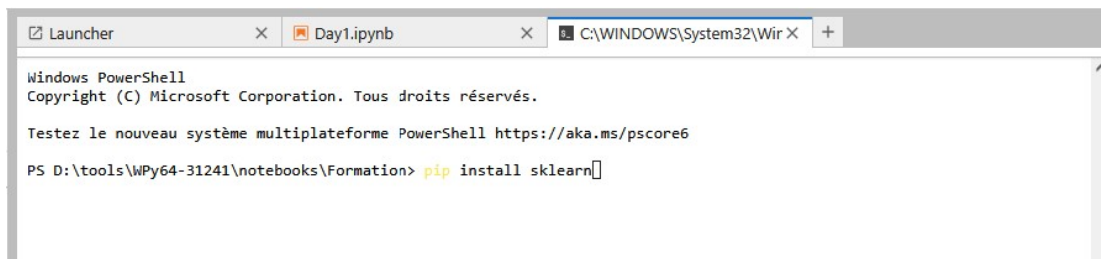
- Run the following command:

```
# Install Python packages
pip install sklearn
```

- Press `y` and enter whenever you see `Proceed [y]/n` from your terminal.
- Close the command window after the installation finishes, log out of Windows, and then log in.

You can navigate to the directory containing the training materials that you downloaded from Github

Once the `index.ipynb` notebook is open, you should see something similar to the following image:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS D:\tools\WPy64-31241\notebooks\Formation> pip install sklearn
```

The root directory `formation` contains five days training program and a dataset directory including all the data used during the training labs.

Open the file `local_install.ipynb` in Jupyter by navigating to the `Formation` folder that we downloaded earlier from github, then click on the `day1` folder.

Jupyter Notebook

This is the actual file that allows you to mix code and text to explain it, notice that the content inside a Jupyter notebook is organized into cells.

Cells can have *inputs* and *outputs*.

There are two main types of cells:

1. Markdown cells

- *Inputs* are written in [markdown](#) and can contain formatted text, images, equations, and more.
- *Outputs* are rendered **in place of the input** when the cell is executed.

1. Code cells

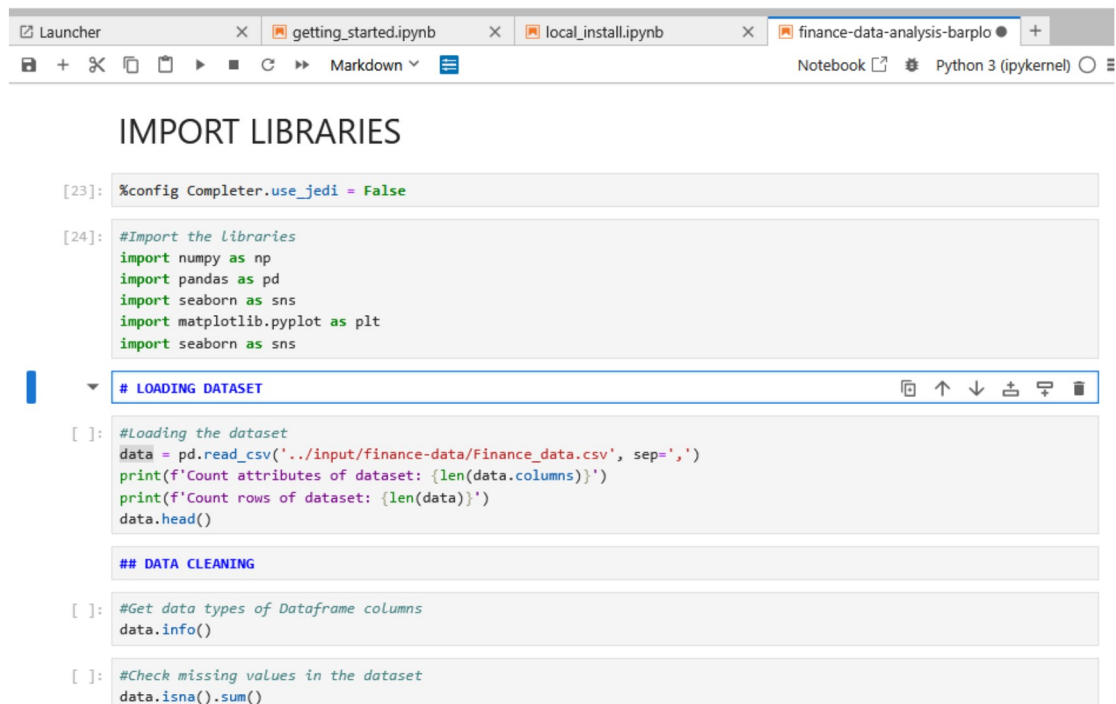
- *Inputs* Contain Python code.
- *Outputs* are placed below the input cell and contain the results generated when the input code is executed.

You can execute a cell by pressing combination of `Ctrl + Enter` (meaning `Ctrl` and `Enter` at the same time)

You can also run a cell and goto the next one by pressing `Shift + Enter`

You can save changes on the notebook by pressing `Ctrl + S`

Below is an image that demonstrates what a Jupyter Notebook looks like:



Notice a few things about this image:

- Code cells that have not yet been executed do not have a number in the `[]:` box and have no corresponding output.
- Run code cells put a `[number]:` to the left of them and, depending on what the code in that cell does, may or may not have an output (message or graphics).
- Run markdown cells are displayed as formatted text, `#` for title, `##` for subtitle.
- `-` as chip and `1.` as number.

Exercise

In the markdown cell bellow write a title1: First Notebook

In the *code* cell below type a quote (`'`), hello world, then another quote (`'`), or type a double quote (`"`), hello world, then another double quote (`"`) and then evaluate the cell

markdown here

In `[]:` *# code here!*