

AI/ML for Climate Workshop

International Livestock Research Institute (ILRI)

hide: - toc


Python Setup & Workflow (Windows • macOS • Linux)

Interactive Setup Guide



Click the **Binder** button above to launch an interactive setup guide where you can test Python installations and environments!

What you'll learn: - Install Python on Windows, macOS, and Linux - Use **conda/mamba** and **pip** effectively - Create & activate virtual environments (`conda` , `venv` , `virtualenv` , `pyenv`) - Install & use **Jupyter Notebooks/Lab** and **VS Code** - Work efficiently in **Google Colab** (GPU/TPU, Drive, packages) - Troubleshoot common setup issues & use cheat-sheets

 **Tip:** Run this notebook cell-by-cell on your machine after installing Python, or open it in **Google Colab** to follow along.

0) Quick Start: Verify Your Current Python & Pip

Use this to see what's already installed.

```
import sys, platform, subprocess, shutil
print("Python executable:", sys.executable)
print("Python version:", sys.version)
print("Platform:", platform.platform())
```

```
def which(cmd):
    return shutil.which(cmd) or "<not found>"

print("pip:", which("pip"))
print("conda:", which("conda"))
print("mamba:", which("mamba"))
print("python:", which("python"))
print("py:", which("py")) # Windows launcher
```

1) Conda Distributions: Anaconda, Miniconda, Miniforge, Mambaforge

Choose one of these Conda-based options:

- **Anaconda** (full): Large installer including hundreds of data-science packages, **Anaconda Navigator** GUI, and `conda`. Uses the **defaults** channel by default. Good for offline/enterprise labs.
- **Miniconda** (minimal Anaconda): Small installer with just `conda` and Python. Uses **defaults** by default; you add packages as needed. Lightweight and flexible.
- **Miniforge** (community): Minimal installer that defaults to the **conda-forge** channel (huge community repo with very fresh builds). Great for scientific/geo/ML stacks.
- **Mambaforge**: Miniforge + `mamba` preinstalled (a much faster solver). Recommended if you want speed and conda-forge by default.
- Choose **one** approach per OS.
- **Miniforge/Mambaforge** is usually the smoothest because it provides prebuilt packages.

Windows

Option A (Recommended): Miniforge (Conda-forge)

1. Download **Miniforge** for Windows from the official repo [Miniforge](#)
2. Run the installer → check **"Add Miniforge to PATH"** and **"Initialize for PowerShell"** if prompted.
3. Open Miniforge terminal.

Verify: Command Prompt/Powershell

```
conda --version
python --version
```

```
conda env list
```

Option B: Anaconda

- Anaconda is a popular open-source distribution of Python and R programming languages.
- It simplifies package management and environment management.
- Designed for data science, machine learning, AI, and scientific computing.
- Comes with a collection of pre-installed libraries like NumPy, Pandas, Matplotlib, SciPy, Scikit-learn, and more.
- Includes tools like Jupyter Notebook, Spyder, and Conda for streamlined development.

Download the Installer

- Visit the official Anaconda [Distribution page](#)
- Download the latest Anaconda installer for Windows.

Launch the Installer

- Navigate to your Downloads folder.
- Double-click the downloaded installer to start the installation process.

Start the Installation

- Click Next on the welcome screen.

Accept the License Agreement

- Read the licensing terms.
- Click I Agree to proceed.
- Tip: It is recommended to select "Just Me" during installation. This ensures Anaconda is installed only for the current user.

Choose Installation Type

- Click Next to confirm your installation type.

Select Installation Folder

- Choose a destination folder for Anaconda.
- Click Next to proceed.

Important: - Avoid installing Anaconda in a directory path that contains spaces or Unicode characters. - Do not install as Administrator unless admin privileges are necessary. - Do not add Anaconda to the PATH environment variable, as this may cause conflicts with other software.

Install Anaconda

- Click Install and wait for the installation to complete.
- Click Next to proceed.

Finish Installation

- After a successful installation, you will see the "Thanks for installing Anaconda" dialog box.

Additional Resources

- For more detailed information, refer to the Anaconda Installation Documentation.

Option C: Python.org Installer

1. Download the latest [Windows installer](#) from python.org.
2. During setup, check "Add Python to PATH".
3. Open **Command Prompt/Powershell**:

```
python --version
pip --version
```

Option D: Install Linux on Windows 11 with Windows Subsystem for Linux (WSL)

- The Windows Subsystem for Linux (WSL) lets developers use both Windows and Linux at the same time on a Windows machine.

1) Go to the search bar on your Windows Desktop and type "Windows Features" 2) Scroll down until you see "Windows Subsystem for Linux" and "Virtual Machine Platform" click the checkbox 3) Click "OK" and Click "Restart Now". when you are ready. 4) Open **Command Prompt/Powershell**:

```
``` wsl --status wsl --update
```

```
```
```

4) When your machine is back, go back to the search bar on your Desktop and search for "Microsoft Store". 5) In the Microsoft Store application, search for "Ubuntu" and "GET" 6) For first time set "user name" and "password"

```
ls
cd ..
cd ~
ls
cd mnt
```

- Install **Miniforge** inside WSL for conda-based workflows.

Linux (Ubuntu/Debian/Fedora)

```
# Ubuntu/Debian
sudo apt update
sudo apt install -y python3 python3-venv python3-pip

# Fedora
sudo dnf install -y python3 python3-pip python3-virtualenv
```

macOS

Option A (Recommended): Miniforge (Conda-forge)

1. Download **arm64** (Apple Silicon) or **x86_64** (Intel) installer.
2. Run the `.pkg` or use the `.sh` installer from Terminal.
3. Restart the terminal.

```
conda --version
python --version
```

Option B: Homebrew + (pyenv or CPython)

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/installation)"
brew install python # or: brew install pyenv && pyenv install 3.12.6
python3 --version
pip3 --version
```

⚠️ Avoid relying on the macOS system Python for development.

2) Package Managers: `conda` vs `pip`

Conda/mamba manage Python *and* non-Python deps (e.g., `gdal` , `geos` , `cuda`). **Pip** installs Python packages from PyPI.

Conda / Mamba Basics

```
conda init                # set up your shell once
conda config --add channels conda-forge
conda config --set channel_priority strict
conda create -n ds-312 python=3.12
conda activate ds-312
conda install numpy pandas jupyterlab
conda list
conda env export > environment.yml
conda env create -f environment.yml
conda remove --name ds-312
conda remove -n ds-312
conda remove
```

Deactivate:

```
conda deactivate
```

Pip Basics

```
python -m pip install --upgrade pip
pip install numpy pandas jupyterlab
pip freeze > requirements.txt
pip install -r requirements.txt
```

3) Create & Activate Virtual Environments

- When installing Python on your machine, you have the option to install it `system-wide` or in a `virtual environment` .
- Installing Python system-wide may seem like the most straightforward approach, it can lead to several problems such as `dependency conflicts` and `version conflicts` .
- Installing Python in a virtual environment provides several benefits such as `Isolation` , `Dependency management` , and `Reproducibility` .
- A `virtual environment` is an `isolated Python environment` that allows you to `manage dependencies` for different `projects` without conflicts.

- Below are step-by-step guides for creating virtual environments using Conda and pip.

A) Conda

```
conda create -n myenv python=3.11
conda activate myenv
conda install numpy
```

Deactivate:

```
conda deactivate
```

Python Environment management with pip

- Python Default Package Manager: pip is the standard/default package manager for Python.
- Lightweight: Designed for Python-only projects and lightweight environments.
- Virtual Environment Support: Works with venv or virtualenv to create isolated environments.
- Ease of Use: Simple syntax for installing and managing packages.
- Community Driven: Extensive library ecosystem supported by the Python community.
- Customizable Requirements: Supports dependency management with requirements.txt files.

B) Built-in `venv` (works everywhere)

```
python -m venv .venv
# Windows (PowerShell)
.venv\Scripts\Activate.ps1
# macOS/Linux (bash)
source .venv/bin/activate
python -m pip install --upgrade pip
pip install numpy
```

Deactivate: `deactivate`

3) Interactive Development Environments

Jupyter Notebooks

Jupyter Notebooks are a versatile tool for professionals, researchers, educators, and learners, providing an interactive and collaborative environment for coding, documentation, and visualization.

1. Interactive Computing

- Real-Time Execution: Write, execute, and modify code interactively in real-time.
- Experimentation: Ideal for experimenting with algorithms, models, and data.
- Visualization: Allows seamless integration of code and visual outputs for data exploration.

2. Documentation

- Comprehensive Notes: Combine code, text (using Markdown), images, and equations in one place.
- Collaboration Ready: Simplifies sharing results and collaborating with others.

3. Reproducibility:

- Self-Contained Workflows: Combine code, data, and outputs in one notebook.
- Ease of Sharing: Facilitates sharing and verification of results.
- Research Expansion: Enables others to build on your work effortlessly.

4. Flexibility:

- Multi-Language Support: Supports multiple languages like Python, R, Julia, and more.

5. Data Visualization:

- Visual Insights: Supports libraries like Matplotlib, Seaborn, Plotly, and more.
- Communicate Results: Create professional charts, graphs, and interactive visualizations.

6. Education:

- Learning Platform: Widely used for teaching programming, data analysis, and machine learning.
- Interactive Learning: Provides hands-on experiences to engage students and reinforce concepts.
- Instructor-Friendly: Allows instructors to create interactive tutorials and demonstrations.

Jupyter: Notebooks & JupyterLab

Install and launch inside your target environment.

Install

```
# Conda
conda install jupyterlab ipykernel

# Pip
pip install jupyterlab ipykernel
```

Launch

```
jupyter lab
jupyter notebook
jupyter-notebook
```

4) Integrated Development Environment (IDE)

- `Integrated Development Environment (IDE)` is a software application that provides comprehensive tools for coding, debugging, and managing Python projects.
- It streamlines development by integrating features like syntax highlighting, code completion, debugging, and version control.

Key Features to Look for in an IDE

- `Code Completion` : Auto-suggests methods, variables, and functions.
- `Debugging Tools` : Step through your code to identify and fix issues.
- `Integrated Terminal` : Run Python scripts directly from the IDE.
- `Version Control` : Built-in support for Git to manage project versions.
- `Plugin Support` : Extend functionality with custom plugins or extensions.

Types of IDEs for Python

These IDEs support multiple programming languages, including Python.

- `PyCharm` : A feature-rich IDE with smart code completion, debugging, and built-in testing tools. Ideal for both beginners and advanced users.
- `Visual Studio Code (VS Code)` : A lightweight, highly customizable IDE with Python extensions for code linting, debugging, and virtual environment support.

- `Spyder` : Popular among data scientists for its integration with scientific libraries like NumPy, Pandas, and Matplotlib. It is also a Python specific IDE.

VS Code (IDE)

1. Install [Visual Studio Code](#).
 2. Install the **Python** extension (and **Jupyter** extension if prompted).
 3. **Select Interpreter:** `Ctrl/Cmd+Shift+P` → *Python: Select Interpreter* → pick your env.
 4. Open a folder → VS Code terminal will use your shell; activate env as usual.
 5. Open or create `.ipynb` files and run cells in-place.
-

6) Google Colab

Start with Colab

- Go to <https://colab.research.google.com>
- **New Notebook** or **Upload** your `.ipynb`
- Runtime → *Change runtime type* → Select **GPU** or **TPU** if needed
- Cloud-Based: No need for local installations; everything runs in the cloud.
- Free GPUs and TPUs: Access hardware accelerators for machine learning and deep learning tasks.
- Interactive Development: Combine code, text (Markdown), and outputs in a single document.
- Integration with Google Drive: Automatically save and access notebooks from your Google Drive.
- Collaboration: Share notebooks and collaborate with others in real-time.
- Rich Visualization: Supports libraries like Matplotlib, Seaborn, and Plotly for creating visualizations.
- Easy Library Installation: Install libraries directly with pip or apt.

1) Access Google Colab - Go to Google Drive. - Sign in with your Google account. If you don't have an account, create one first.

2) Create a New Folder - In Google Drive, click the New button on the left sidebar. - Select New Folder and give it a meaningful name (e.g., Colab-Projects). - Click Create to make the folder.

3) Create a New Google Colab File - Navigate to the newly created folder in Google Drive. - Right-click inside the folder and select More > Google Colaboratory. - A new .ipynb file will open in a new browser tab.

4) Mount Google Drive in Colab - To access files in your Google Drive from the Colab notebook, mount the drive:

```
```python
from google.colab import drive
drive.mount('/content/drive')
```

### Save Notebook to Drive
- File → Save a copy in Drive
- Or `File → Download` as `.ipynb` to your machine
```

Replace 'Colab-Projects' with your folder name

```
import os
os.chdir('/content/drive/MyDrive/Colab-Projects')
```

Unmount Drive (Optional)

```
drive.flush_and_unmount()
```

1) Configure the Runtime - Click on Runtime > Change Runtime Type. Choose: - Hardware Accelerator: - None for CPU-only tasks. - GPU for accelerated computations (e.g., TensorFlow, PyTorch). - TPU for advanced deep learning workloads. - Runtime Type: Choose Python 3.

Exercise

1) Installing Anaconda 2) Creating a Conda Environment named 3) Activate the environment 4) Install the following packages: NumPy, Pandas, Matplotlib 5) Verify the installed libraries and their versions 6) Deactivate the environment 7) Get familiar with Jupyter Notebook, VSCODE, Google Colab