

Practical Setup: Conda, IDE, Jupyter & ML Libraries

1. What You Will Learn Today

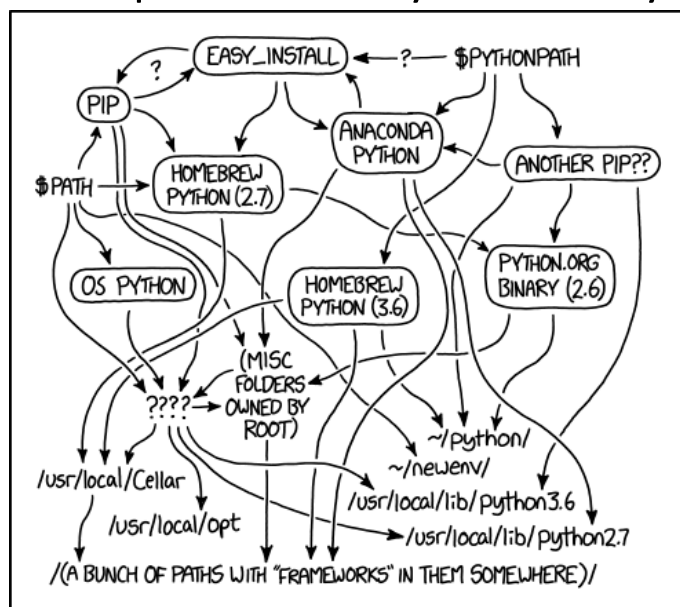
- Install Conda
- Create and activate a virtual environment
- Install Jupyter Notebook
- Install required ML libraries
 - NumPy
 - Matplotlib
 - Scipy
 - Scikit-learn
- Run your notebook

2. Why do we need Conda?

- Different projects require different library versions
- Installing everything globally causes conflicts
- Conda keeps projects separate and safe

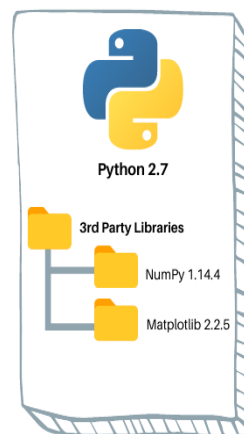
Key idea:

One computer → many isolated Python environments

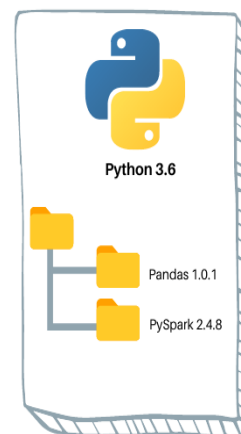


MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

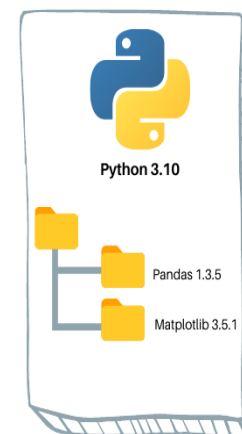
Virtual Environment 1



Virtual Environment 2



Virtual Environment 3



3. Without vs With Conda

Installing Conda, Jupyter Notebook, and Libraries:

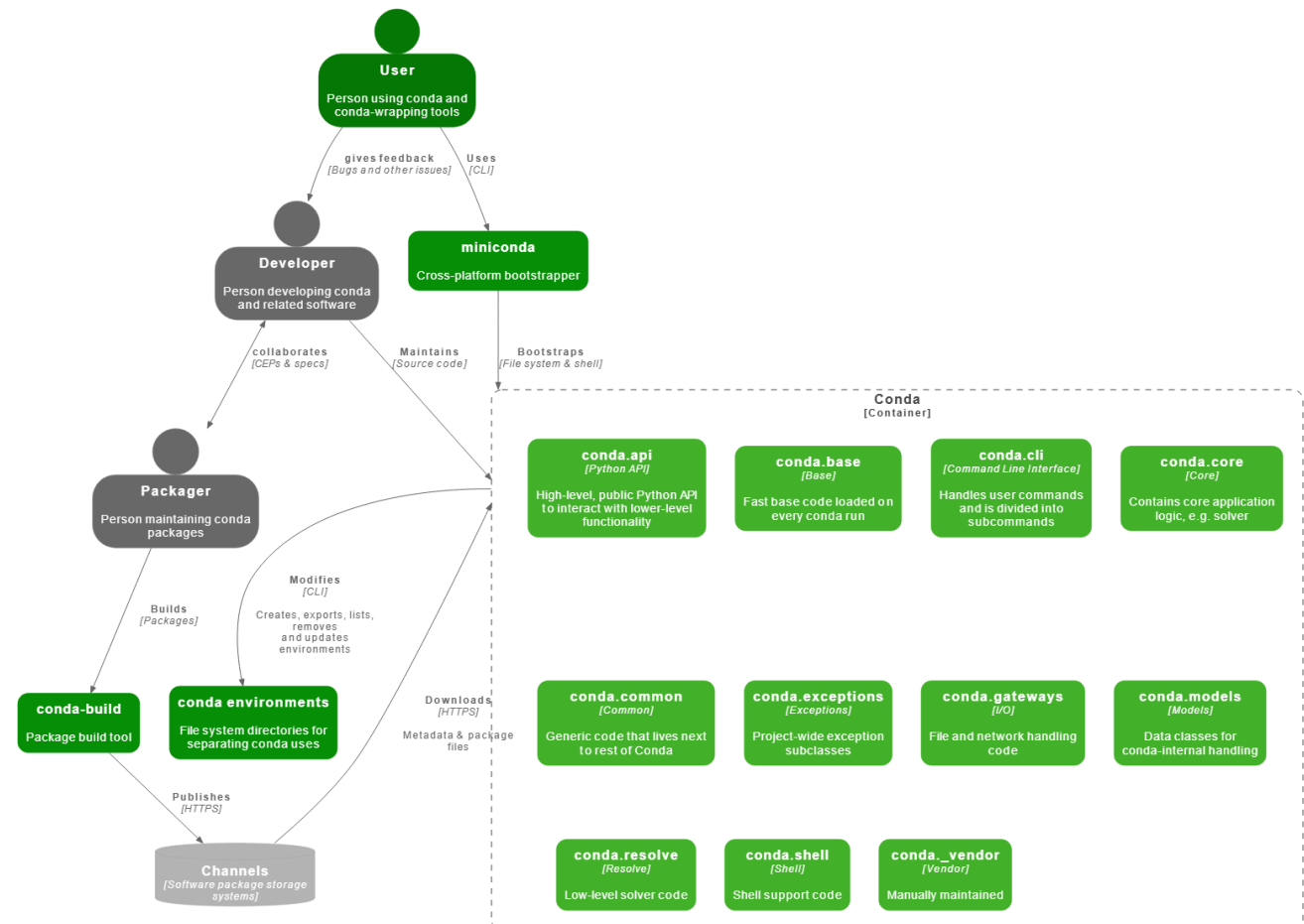
Without Conda:

Python

- └─ numpy (old)
- └─ numpy (new) ✗ conflict
- └─ torch (fails)

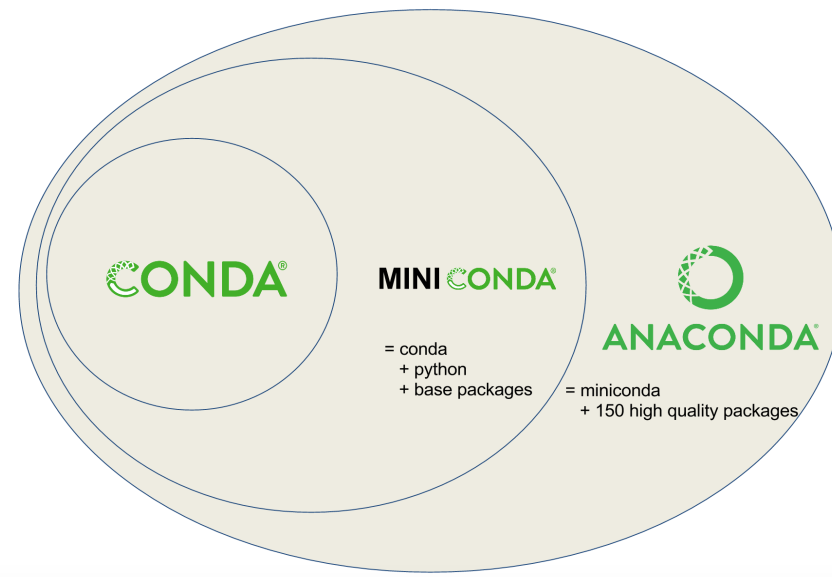
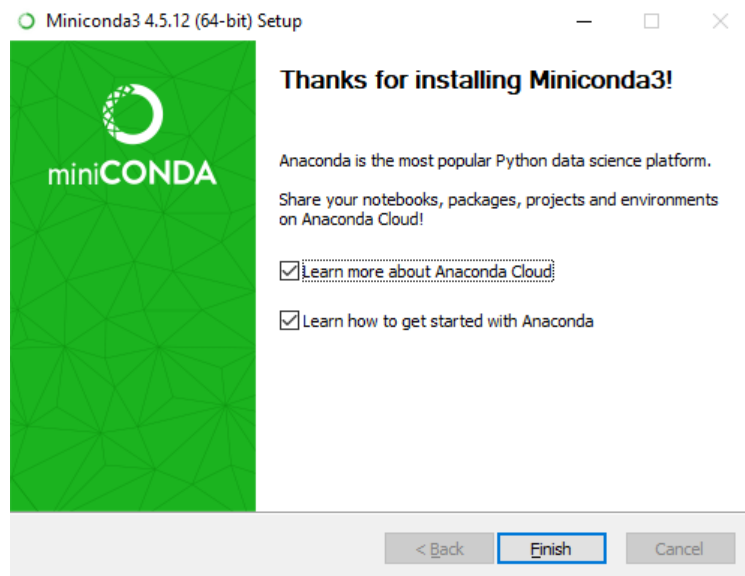
With Conda:

Environment A	Environment B
Python 3.9	Python 3.11
numpy 1.23	numpy 1.26
torch 2.0	torch 2.2



4. Install Miniconda (Windows)

- Download Miniconda (Python 3, 64-bit) from:
- <https://docs.conda.io/en/latest/miniconda.html>
- Install and restart terminal
- Add Conda to PATH



- Or Search for “Anaconda Prompt” Run:

```
conda --version
```

5. Install Miniconda (macOS/Linux)

- Download the installer
- Follow standard install steps
- Open Terminal
- Check installation:

```
conda --version
```

6. Create and activate a new environment

Create an environment called **ml**:

- `conda create -n ml python=3.10`

Activate it:

- `conda activate ml`
- You should now see (ml) in the terminal
- `conda deactivate`

7. Integrated Development Environment (IDE)

- IDE Choices:
 - **PyCharm:**
 - Robust features for development
 - Easy configuration of project interpreters (link to conda env)
 - **VS Code:**
 - Lightweight and popular
 - Great Python extensions and terminal integration
 - **Notebooks (e.g. Jupyter, Colab):**
 - Interactive and useful for exploration

Important: final submissions must be .py files

8. Recap and key takeaways

- Use virtual environments to avoid dependency conflicts.
- Miniconda is a lightweight tool for managing these environments.
- pip remains your go-to tool for package installation within environments.
- Configure your IDE (PyCharm or VS Code) correctly for your projects.
- **Always submit a .py file for coursework.**

9. Creating a Conda Environment from environment.yml

What is **environment.yml**?

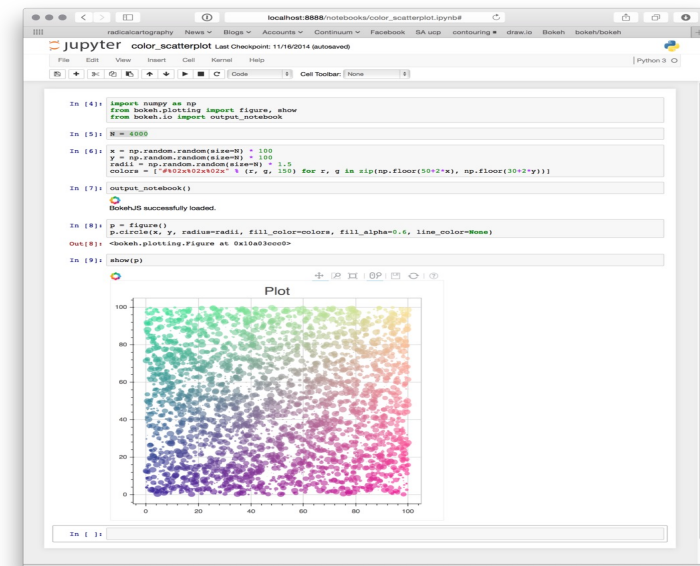
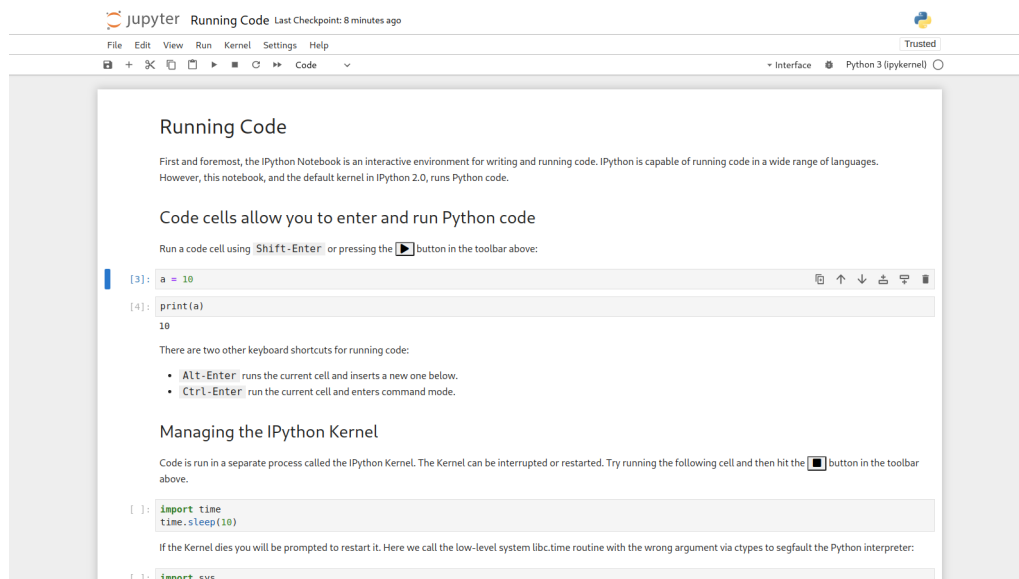
A file that defines a reproducible software environment, including: environment name, Python version, required packages (Conda + pip)

How to Create the Environment

- Open **Anaconda Prompt** and navigate to the folder containing environment.yml `cd path/to/project`
- Create the environment: `conda env create -f environment.yml`
- Activate it: `conda activate coursework_env`

10. What is Jupyter Notebook

- Interactive coding environment
- Code + text + figures in one place



11. Install and launch Jupyter Notebook

Make sure the **ml** environment is active:

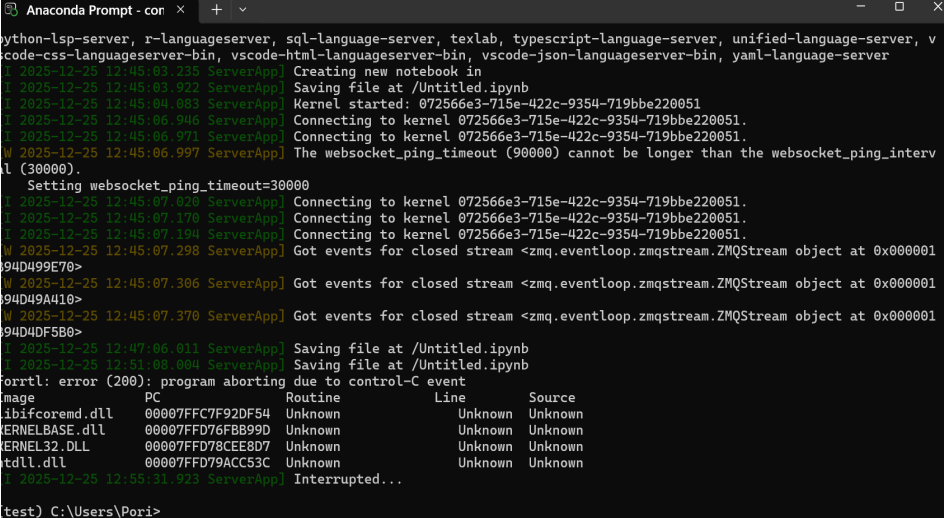
- `pip` install notebook
- `jupyter` notebook

Browser opens automatically

Runs locally on your machine

- Shortcut to stop Jupyter Notebook

Ctrl + C



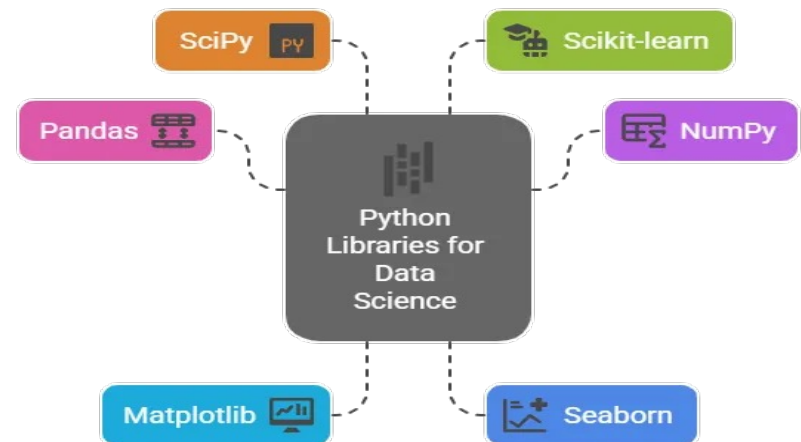
```
Anaconda Prompt - cor x + v
python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, v
code-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yamll-language-server
2025-12-25 12:45:03.215 ServerApp] Creating new notebook in
2025-12-25 12:45:03.222 ServerApp] Saving file at /Untitled.ipynb
2025-12-25 12:45:04.083 ServerApp] Kernel started: 072566e3-715e-422c-9354-719bbe220051
2025-12-25 12:45:06.946 ServerApp] Connecting to kernel 072566e3-715e-422c-9354-719bbe220051.
2025-12-25 12:45:06.971 ServerApp] Connecting to kernel 072566e3-715e-422c-9354-719bbe220051.
2025-12-25 12:45:06.997 ServerApp] The websocket_ping_timeout (90000) cannot be longer than the websocket_ping_interv
l (30000).
Setting websocket_ping_timeout=30000
2025-12-25 12:45:07.020 ServerApp] Connecting to kernel 072566e3-715e-422c-9354-719bbe220051.
2025-12-25 12:45:07.170 ServerApp] Connecting to kernel 072566e3-715e-422c-9354-719bbe220051.
2025-12-25 12:45:07.194 ServerApp] Connecting to kernel 072566e3-715e-422c-9354-719bbe220051.
2025-12-25 12:45:07.298 ServerApp] Got events for closed stream <zmq.eventloop.zmqstream.ZMQStream object at 0x000001
94D499E70>
2025-12-25 12:45:07.306 ServerApp] Got events for closed stream <zmq.eventloop.zmqstream.ZMQStream object at 0x000001
94D49A410>
2025-12-25 12:45:07.370 ServerApp] Got events for closed stream <zmq.eventloop.zmqstream.ZMQStream object at 0x000001
94D4DF5B0>
2025-12-25 12:47:06.011 ServerApp] Saving file at /Untitled.ipynb
2025-12-25 12:51:08.004 ServerApp] Saving file at /Untitled.ipynb
KeyboardInterrupt: program aborting due to control-C event
Image PC Routine Line Source
libifcoremd.dll 00007FFC7F92DF54 Unknown Unknown Unknown
KERNELBASE.dll 00007FFD76FB899D Unknown Unknown Unknown
KERNEL32.DLL 00007FFD78CEE8D7 Unknown Unknown Unknown
stdll.dll 00007FFD79ACC53C Unknown Unknown Unknown
2025-12-25 12:55:31.923 ServerApp] Interrupted...
(test) C:\Users\Pori>
```

12. Install Required ML Libraries

- `pip` install numpy matplotlib scipy scikit-learn

What Each Library Is For

- NumPy → arrays & maths
- Matplotlib → plots
- Scipy → numerical integration & linear algebra & signal and image processing
- Scikit-learn → ML algorithms
- Jupyter → interactive interface



13. Verify Installation in Jupyter

- Create a new notebook and run:
- `import numpy`
- `import matplotlib.pyplot as plt`
- `Import scipy`
- `import sklearn`
- `print('Setup successful')`

14. Common Problems & Fixes

- Command not found → restart terminal
- Module not found → activate environment again
- Wrong Python → check (ml) is active

15. Final Checklist

- Conda installed
- Environment (ml) created and activated
- Jupyter opens correctly
- Libraries import without errors

16. Simple code

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("\nType of X is:", type(X))
print("\nFirst 5 rows of X:\n", X[:5])
```