Installing Anaconda

Anaconda is a popular Python distribution designed for data science and machine learning. It comes pre-installed with essential libraries like NumPy, pandas, and scikit-learn, along with tools like Jupyter Notebook. This makes it easy to set up a consistent environment without worrying about dependencies or compatibility issues. It’s beginner-friendly and saves time by simplifying package management and project setup.

How to Install Anaconda

1. **Download Anaconda:** Go to https://www.anaconda.com/products/distribution. It might ask you for your email but there is also a “Skip Registration” button to directly download anaconda right away without email.
2. **Choose Your OS:** Select your operating system (Windows, macOS, or Linux).
3. **Download the Installer:** Click the appropriate download button (64-bit recommended).
4. **Run the Installer**
    1. **Windows/macOS**: Open the downloaded file and follow the setup instructions.
    2. **Linux**: Dowload the bash file and run it in terminal like this:

        ```
        bash Anaconda3-*.sh
        ```

        Dont forget to replace Anaconda3-* with the correct filename
5. **Verify Installation:** Open terminal or Anaconda Prompt and type:

    ```
    conda --version
    ```

You’re done! It should output the version of conda you just installed

Anaconda vs Miniconda

Miniconda is a lightweight Python distribution with just conda and a few basics, ideal for custom setups. Anaconda is a full package with Python and popular data science libraries pre-installed, great for getting started quickly. Choose Miniconda for flexibility, and Anaconda for convenience.

1. Miniconda:

- **Lightweight version** of Anaconda.
- Comes with **only** `conda`, `pip`, and a few essential packages.
- When you run `conda list` after installing Miniconda, packages like **NumPy will not be present** unless you install it manually using:

  ```
  conda install numpy
  ```

2. Anaconda:

- **Full distribution** with **hundreds of packages** for data science and machine learning.
- Includes **NumPy** and many other popular libraries (like pandas, matplotlib, etc.).
- After installing Anaconda, running `conda list` will **show NumPy** by default.

Summary: Miniconda is minimal and does not come with NumPy, while Anaconda is a larger package that includes it.

Adding conda to Path (Windows)

When you install **Anaconda** or **Miniconda**, the `conda` command-line tool is available in the terminal or shell where you ran the installation script. However, it may not work in other terminals or shells unless you add it to your **environment variables**. To **add `conda` to your environment** (so it works in any terminal or shell), follow these steps based on your operating system:

You can run this command in **Anaconda Prompt** to initialize `conda` for **Command Prompt (cmd)** or **PowerShell**:

```
conda init
```

- This modifies your shell's configuration to recognize `conda`.
- Restart your terminal after running the command.
- The process is automatic and you should be able to use conda from any directory in the terminal.

Manually (If Needed): Add these paths to the **System Environment Variables**:

1. Open:
    1. **Win + R** → Type `sysdm.cpl` to open system properties → Click **Environment Variables**.
2. Find and edit the `Path` variable in **System Variables**.
3. Add these entries:

    ```
    C:\Users\<YourUsername>\Anaconda3\Scripts
    C:\Users\<YourUsername>\Anaconda3\bin
    ```

    Dont forget to replace "<YourUsername>" with your actual username

To verify:

```
conda --version
```

Understanding Conda Workflow

What is Conda?

Conda is a **package manager** and **environment manager** for Python (and other languages). Think of it like a **recipe manager** for your projects!

The Recipe Analogy:

Imagine you're making **mashed potatoes**. The type of **potato** you use changes the taste – in the same way, **package versions** change how your code works.

For example:

- **Python 3.11** is like using a **Shimla Aloo**.
- **Python 3.8** is like using a **Kufri Sindhuri Aloo**.

Conda lets you **control the ingredients** (package versions) for each project.

---

1. Managing Environments with Conda

By default, you are in the **base** environment.

Check Available Environments

Run the following command to list the available environments

```
conda env list
```

2. Create a New Environment

Use this command to create a new conda environment

```
conda create -n myenv
```

Tip: You can create an environment with a specific Python version:

```
conda create -n myenv python=3.11
```

3. Activate an Environment

```
conda activate myenv
```

4. Install Packages (Example: from conda-forge Channel)

```
conda install -c conda-forge numpy
```

- `-c` flag specifies the **channel** (a source for packages).
- **conda-forge** is a popular community-maintained channel with cutting-edge packages.

5. Deactivate the Environment

```
conda deactivate
```

Summary

- **Conda** is a package and environment manager for Python.
- It lets you manage different environments for different projects.
- You can create, activate, and deactivate environments using `conda`.
- Use `conda install` to add packages to an environment.
- **Conda-forge** is a community-maintained channel with many packages.

- **Tip:** Always activate an environment before installing packages.

Anaconda Navigator - Quick Tour

What is Anaconda Navigator

Anaconda Navigator is a graphical user interface (GUI) that comes with the Anaconda distribution. It lets you launch tools like Jupyter Notebook, Spyder, and manage environments and packages without using the command line.

To open Navigator in:

- **Windows:** Search for **Anaconda Navigator** in the Start menu.
- **macOS/Linux:** Run `anaconda-navigator` in the terminal.

It's beginner-friendly and useful for managing your data science workflow visually.

Jupyter Notebook vs. JupyterLab

Jupyter is an open-source tool for creating and sharing documents that contain **live code**, **visualizations**, **equations**, and **text**.

What is Jupyter Notebook?

- A simple, **interactive environment** to write and run code.
- Each document (called a **notebook**) is saved as a `.ipynb` file.
- Useful for **quick prototyping**, **data exploration**, and **teaching**.
- A **cell** in Jupyter Notebook is a container where you can write and run code, markdown, or raw text.

Start Jupyter Notebook by typing:

```
conda install jupyter
```

followed by:

```
jupyter notebook
```

What is JupyterLab?

JupyterLab is the **next-generation interface** for Jupyter, offering a **more advanced** and **flexible** workspace.

Why We Use JupyterLab:

- **Multi-Panel Interface**: Open notebooks, terminals, text editors, and outputs **side by side**.
- **Better Navigation**: File browsing and workspace management are easier.
- **Extensions**: Customize with themes and advanced plugins.
- **Integrated Tools**: Combine notebooks with terminals and markdown in one window.

Start JupyterLab:

```
conda install jupyterlab
jupyter lab
```

Key Differences:

Feature	Jupyter Notebook	JupyterLab
Interface	Single document view	Multi-tab, multi-panel
Customization	Limited	Extensive via extensions
Performance	Lightweight	Slightly heavier
Use Case	Quick tasks, tutorials	Large projects, workflows

When to Use What:

- **Jupyter Notebook**: Simple analysis, tutorials, and quick checks.
- **JupyterLab**: Complex projects, interactive dashboards, and better organization.

We will use JupyterLab because it provides:

1. **Better Workflow** – Manage multiple notebooks and files in one view.
2. **Scalability** – Ideal for **data science** and **machine learning** projects.
3. **Future-Proofing** – JupyterLab is actively developed and might eventually **replace** Jupyter Notebook (My personal opinion).

As already told you in the video, one of the organizers of the conference I attended in Orlando told me that Jupyter Notebook is a blog where you can execute code. It is an information-rich coding interface. I like this way of looking at it. Hope you are enjoying this course so far!