Advanced Hydrological Modeling with Machine Learning and Earth Observation
Chair of Hydrology and River Basin Management
TUM School of Engineering and Design
Technical University of Munich

Download and Setup of Anaconda and Jupyter Notebook

Author: Moritz Wirthensohn[1]

[1]moritz.wirthenshohn@tum.de

Last Update: April 17, 2024

Contents

1 Introduction

Before we delve into the world of Python and Jupyter Notebook, let's familiarize ourselves with a couple of terms that you will encounter frequently:

- **Module**: In Python, a module is a single Python file that contains a collection of functions, classes, and variables (e.g., $math.py$). This file can be imported and used in other Python programs and typically focuses on a specific functionality.
- **Package:** In Python, a package is a collection of related modules that work together to provide certain functionality. These modules are contained within a folder and can be imported just like any other modules. This folder will often contain a special $\_\_init\_\_.py$ file that tells Python it's a package, potentially containing more modules nested within subfolders.
- **Library:** A library is an umbrella term referring to a reusable chunk of code. Usually, a Python library contains a collection of related modules and packages. This term is often used interchangeably with "Python package" because packages can also contain modules and other packages (subpackages). However, it is often assumed that while a package is a collection of modules, a library is a collection of packages.
- **Virtual Environment:** Python virtual environments are isolated Python environments that allow you to install and manage packages independently for different projects. This is useful when you have multiple projects with different dependencies or when you want to avoid conflicts between packages. Virtual environments ensure that your project's dependencies are isolated and don't interfere with other projects on your system.

Now that we understand these terms let's explore Python and Jupyter Notebook:

Python is a high-level programming language known for its simplicity and readability. In the context of hydrological modeling, Python serves as a valuable tool for analyzing complex data sets, implementing mathematical models, visualizing results, etc.. One of the key strengths of Python is its vast collection of libraries and packages tailored for data analysis and machine learning. These provide pre-written code for performing common tasks, such as numerical computation, data manipulation, and visualization. Some of the most used libraries are:

- **NumPy:** NumPy is a powerful library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.
- **pandas:** pandas is a library for data manipulation and analysis in Python. It offers data structures like DataFrame and Series, which allow you to work with structured data easily. pandas is commonly used for tasks such as cleaning data, exploring datasets, and performing statistical analysis.
- **matplotlib:** matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. It provides a MATLAB-like interface for generating plots, charts, histograms, and other graphical representations of data.

To write, edit, debug, and execute Python code efficiently, developers and engineers often use Integrated Development Environments (IDEs). IDEs are software applications designed to streamline and enhance the development process. Some popular Python IDEs include: Visual Studio Code, Spyder and PyCharm. On the other hand, Jupyter Notebook is a web-based interactive computing environment that allows you to create and share live code, visualizations, equations, and text blocks. Unlike traditional IDEs, Jupyter Notebook operates as a computational notebook, enabling users to run code in small sections called "cells" and see the output immediately below each cell. This makes it easier to follow the development process. The ability to combine code, text, and visualizations in a single document also makes Jupyter Notebook a powerful tool for sharing and communicating your work with others.

In the following sections of this tutorial, you will learn how to download, install, and use Jupyter Notebook as part of the Anaconda distribution, which simplifies the management of Python packages and environments.

2 Anaconda

2.1 Overview

Anaconda is a comprehensive distribution of Python and R programming languages, along with hundreds of pre-installed scientific computing packages and libraries. It offers several key features that make it invaluable for managing Python environments and packages:

- **Package Manager:** Anaconda simplifies the process of installing, managing, and updating packages in Python. It automatically handles dependencies, ensuring that the required libraries are installed along with the package you want to use. This makes it easier to set up your development environment and avoid compatibility issues between different packages.
- **Environment Manager:** One of the most significant advantages of Anaconda is its support for creating isolated Python environments. These environments allow you to install and manage different sets of packages for different projects without conflicts. You can create an environment with specific versions of Python and required libraries, ensuring that your project remains consistent and reproducible over time. This is especially useful when working on multiple projects or collaborating with others who may have different software requirements. For example, you can easily set up separate environments tailored to specific tasks, such as geospatial data processing and machine learning.
- **Distribution of Python and R Programming Languages:** Anaconda not only provides Python but also includes support for the R programming language. This makes it a comprehensive platform for data science and statistical analysis, as it allows users to seamlessly switch between Python and R depending on their preferences and project requirements.

2.2 Download and Installation

Installing Anaconda is a straightforward process, and it can be done on Windows, macOS, and Linux operating systems. Below is a step-by-step guide explaining each step of the downloading and installation process for windows system, but it should be similar for the other operating systems as well:

1. **Download Anaconda:**
   - Visit the official Anaconda website at `https://www.anaconda.com/download`
   - At the top of the page, provide your email and click $Submit$
   - The link in the email will lead you to the download page (Figure 1). Here, select the installer suitable for your operating system. For macOS you should also select the graphical installer
   - Save the installer file to your computer

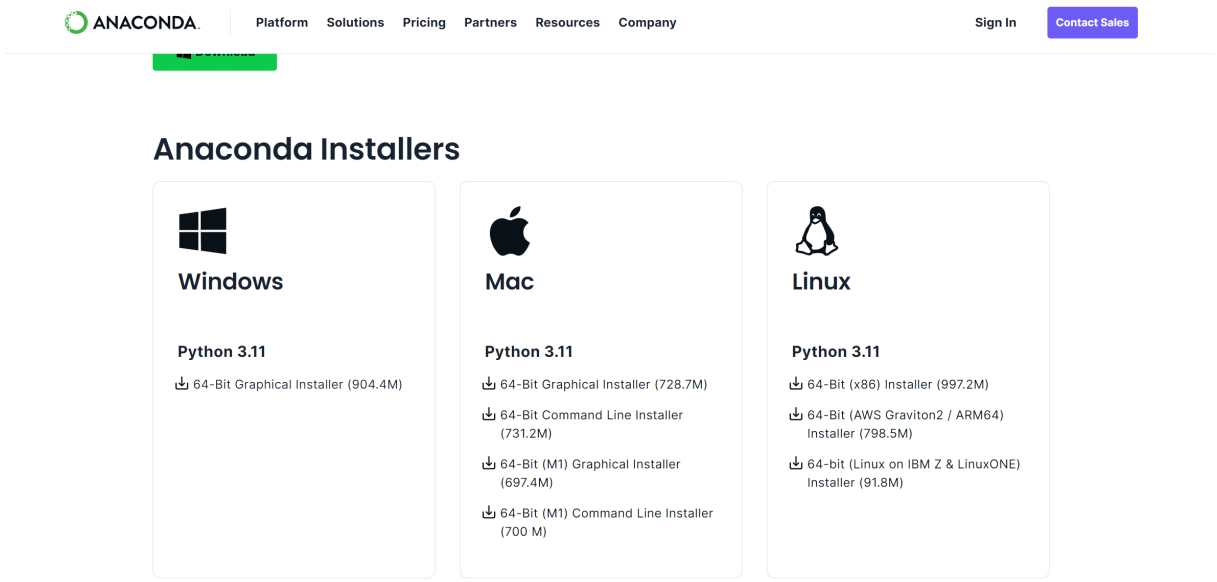

Figure 1: Download options for Windows, macOS and Linux

2. **Run the Installer:**
    - Double click on the installer file to run it
    - The Anaconda installer window will appear. Click "Next" and agree to the License Agreement
    - Install for "Just Me" and select the destination folder where Anaconda will be installed
    - Click "Next". Leave the advanced options as they are. Click "Next"
    - The installer will now begin installing Anaconda on your system. This might take a while. Once the installation is finished, click "Next"
    - Installation completed

Once the installation is finished, you have two options to utilize Anaconda: either through the Anaconda Navigator, which offers a graphical user interface, or via the command line. Both methods will be detailed in the following sections.

2.3 Using Anaconda Navigator

After completing the installation, locate and launch "Anaconda Navigator" to initiate the application (see Figure 2). If updates are available, you can execute them.

Once launched, the "Home" page (Figure 3) will be displayed, featuring tabs like "Home", "Environments", "Learning", and "Community" on the left. The "Home" tab shows applications installed in the current virtual environment, initially limited to the "base (root)" environment. This serves as Anaconda's base environment with essential packages pre-installed, including applications like Jupyter Notebook. Additional applications can be installed within this base environment. Upon creating other environments, you can

switch between them using the drop-down menu (highlighted in red in Figure 3) to manage your applications.

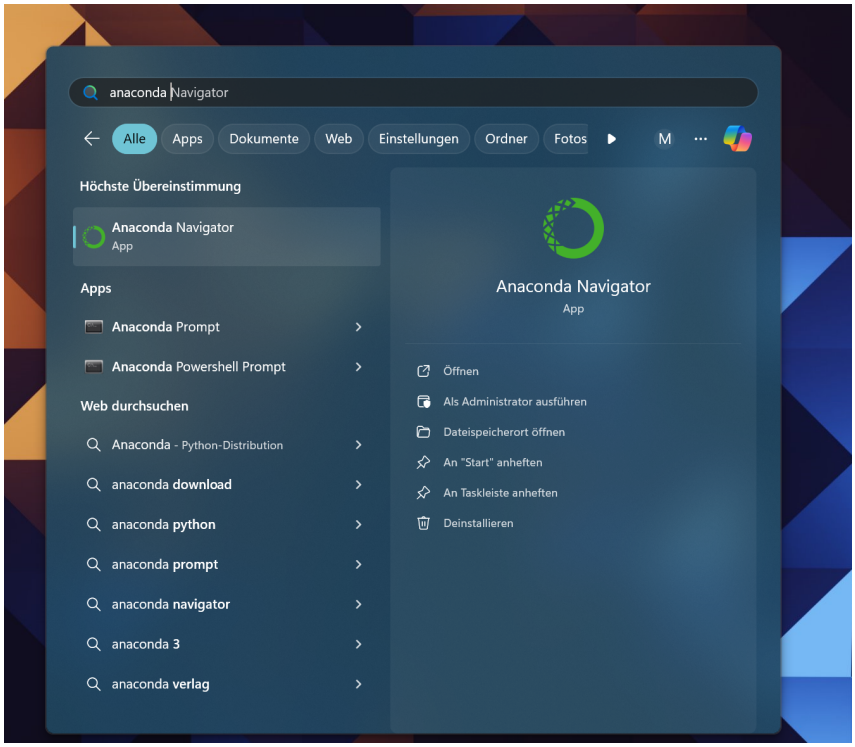

Figure 2: Run the Anaconda Navigator application

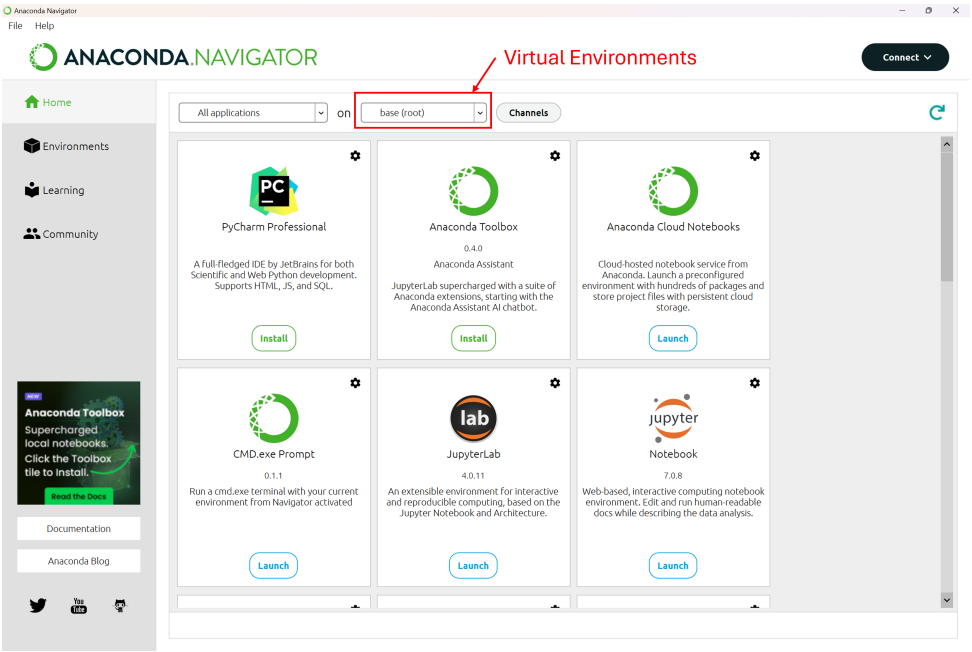

Figure 3: Anaconda Navigator start page

Navigate to the "Environments" tab on the left

In Figure 4, note that only the "base (root)" environment is listed, as no additional

environments have been created yet. On the right, all pre-installed packages, including their descriptions and versions, such as Python, matplotlib, and numpy are shown. Utilize the search bar to find installed packages or select "Not installed" to search for new ones. However, it's not recommended that all packages be installed in the base environment. Instead, follow these steps to create a new environment and install packages there:

1. At the bottom, click on "Create"
2. Give the new environment a proper name (avoid spaces in the name and use underscores instead)
3. Under Packages select "Python" and its version (you can leave the default one)
4. Click "Create"

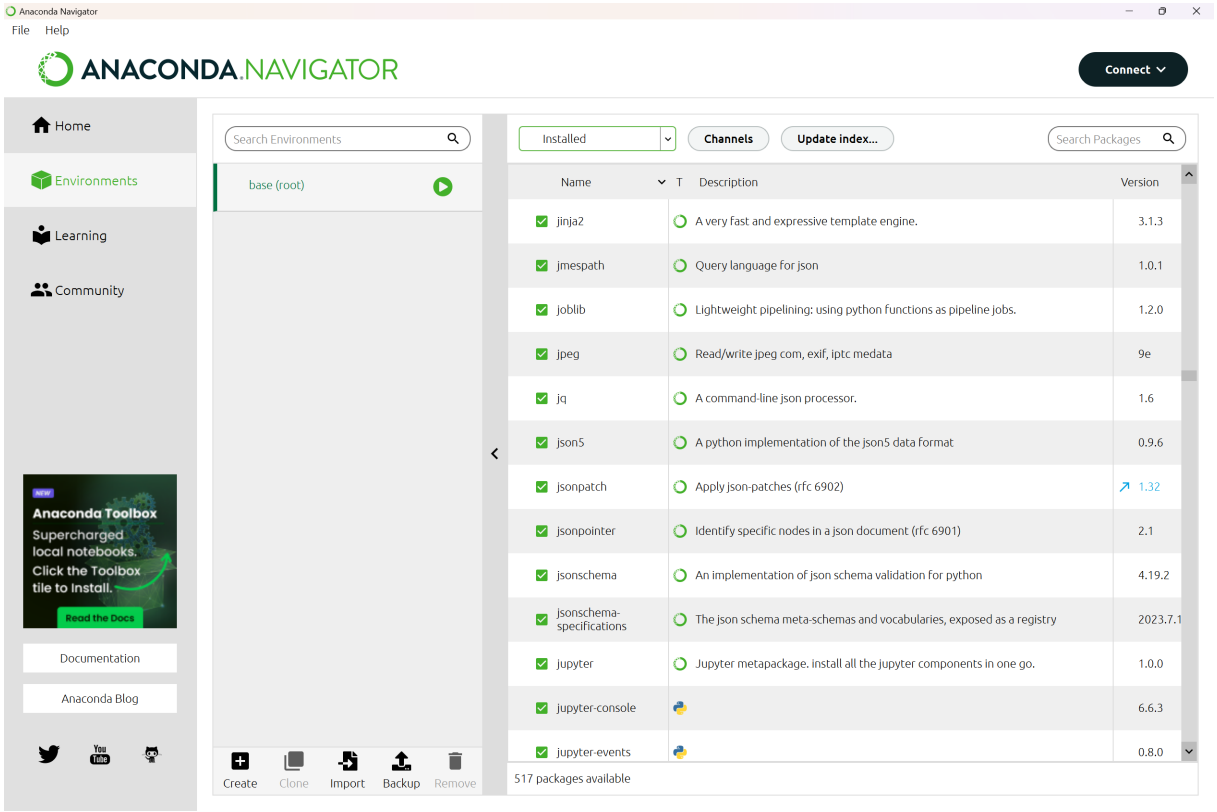

Figure 4: Environments tab in Anaconda Navigator

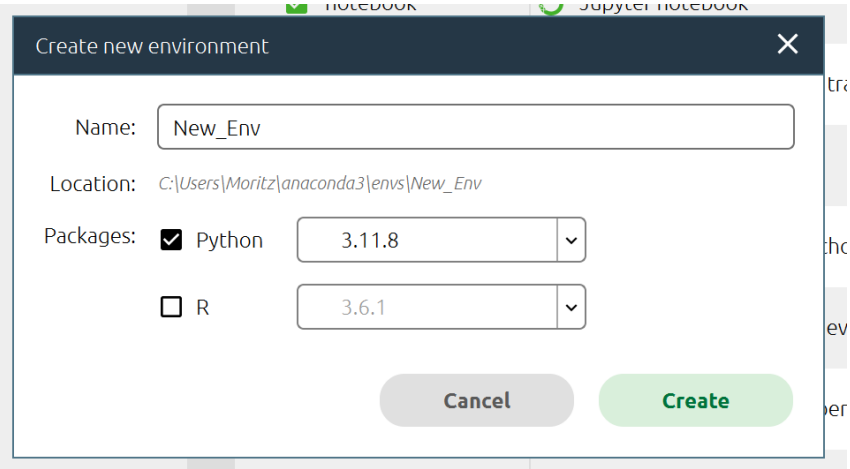

Figure 5: Creating a new environment

A new environment with your created name should appear on the list, and you will see the installed packages on the right. We only selected Python as the package to be installed during the creation process. However, as you can see, Anaconda installed some further packages along with Python, which are necessary to execute Python. This is one of Anaconda's main benefits, eliminating the need to manually install essential packages.

To install additional packages in the new environment, the Navigator primarily searches and installs from Anaconda's free public repositories, known as defaults. However, numerous other package channels are available, such as the popular "conda-forge" channel, which we will add first.

1. In the "Environments" tab navigate to the "Channels" section at the top right
2. Click on "Add..."
3. Enter "conda-forge" into the provided textbox and press enter
4. Click "Update channels"

With the conda-forge channel added Navigator will now search for and install packages from this source. Now, we will continue to install additional packages in our newly created environment:

1. Ensure that the correct environment is selected from the list
2. Choose "Not installed" from the drop-down menu
3. Search for "numpy"
4. Select the package named "numpy"
5. Click "Apply" at the bottom
6. A window will appear displaying additional packages and dependencies to be installed alongside numpy

7. Confirm by clicking "Apply" again

After selecting and installing your desired packages, be patient as it may take some time for them to download and install. You can proceed to add more commonly used packages like "matplotlib" and "pandas" using the same method. This time, you can streamline the process by selecting multiple packages at once for installation instead of doing it individually.

However, don't feel compelled to install all these packages immediately. As you progress over the next few weeks, you'll identify which packages are essential for your tasks. Install them as needed.

Additionally, once you've created a new environment, you can easily duplicate it. Simply select the environment you wish to clone in the "Environments" tab and click "Clone" at the bottom. Give the new environment a name, and you'll have a replica with the same packages installed, ready for further installations. Similar, you can remove the environment by selecting the environment and clicking the "Remove" button

Now, return to the "Home" tab, and select your new environment from the drop-down menu. You might notice that certain applications previously installed in the base environment are missing. One such example is Jupyter Notebook. Follow these steps to install it:

1. Navigate back to the "Environments" tab
2. Ensure your newly created environment is selected
3. Choose "Not installed" from the drop-down menu and search for "nb_conda"
4. Select "nb_conda" package
5. Click "Apply", then confirm by clicking "Apply" again

Now, when you return to the "Home" tab and select "Installed Applications" from the drop-down menu, you should see Jupyter Notebook listed with a blue "Launch" button. Details on setting up Jupyter Notebook and your conda environment within it will be provided in Section 3.
Important: When you create a new environment and want to use it in Jupyter Notebook, you always have to install the "nb_conda" package. Or, there are other ways to install and launch Jupyter Notebook, but installing "nb_conda" comes with the relevant dependencies and makes it easier.

2.4 Using Anaconda Command Line

An alternative method for utilizing Anaconda is through the command line interface, without using the Navigator's graphical interface. Windows users can access this by searching for "Anaconda Prompt" (Figure 6), while macOS and Linux users can use the "Terminal".

Figure 6: Run the Anaconda Prompt

When you open the Anaconda Command Line or Terminal, you'll see the following structure: Initially, the name of the currently activated environment is displayed (in this instance, it's the "base" environment). Following that, you'll find the current directory path.

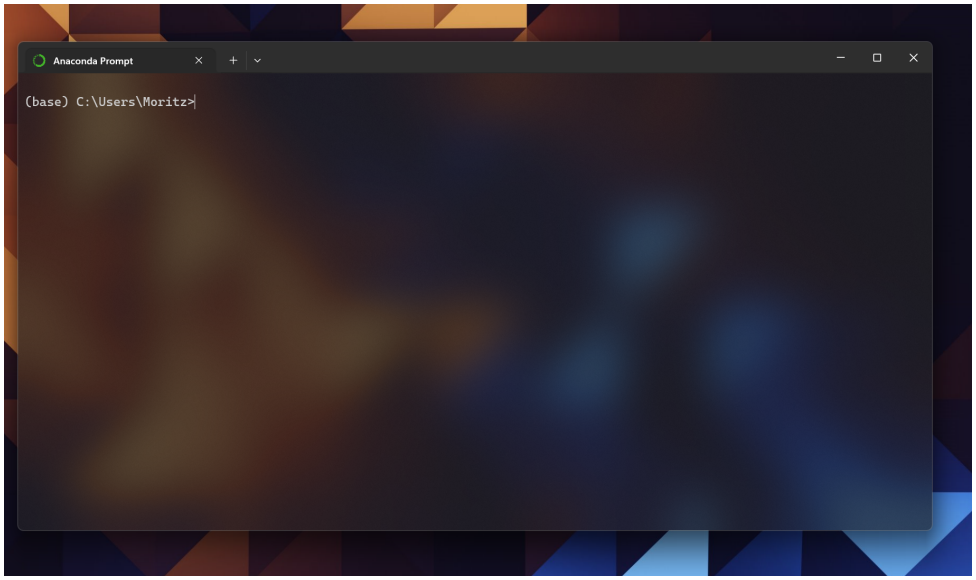

Figure 7: Start screen Anaconda Command Line

To create a new environment via the command line, use the following command:

conda create --name <new_environment_name>

Replace <new_environment_name> with an appropriate name.
Alternatively, you can directly install specific packages during the creation process:

conda create --name <new_environment_name> python numpy pandas

This will install Python, numpy, and pandas directly into your new environment.
Once the new environment is created, you need to activate it to utilize and install additional packages:

conda activate <new_environment_name>

Upon activation, you'll notice the command line displaying the name of the activated environment instead of "base". To deactivate the current environment and return to the base environment, use:

conda deactivate

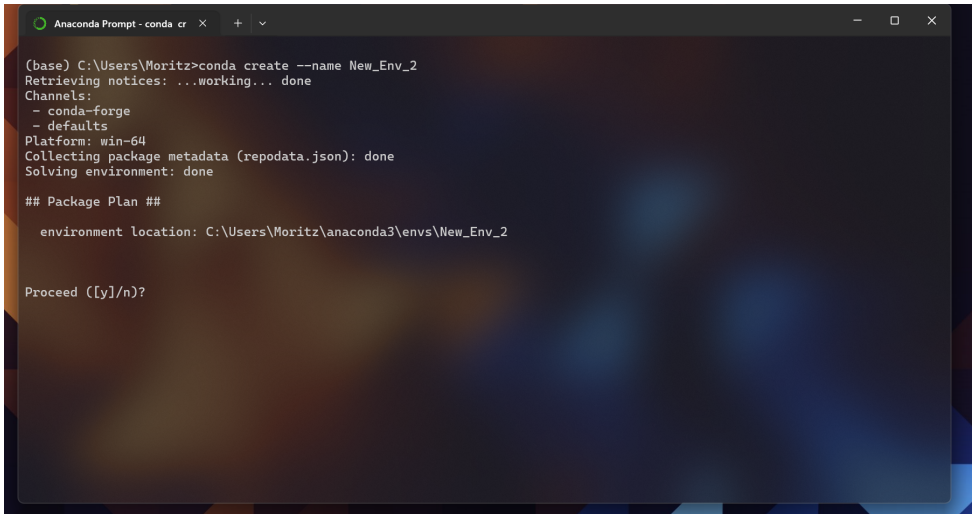

Figure 8: Creating a new environment via command line

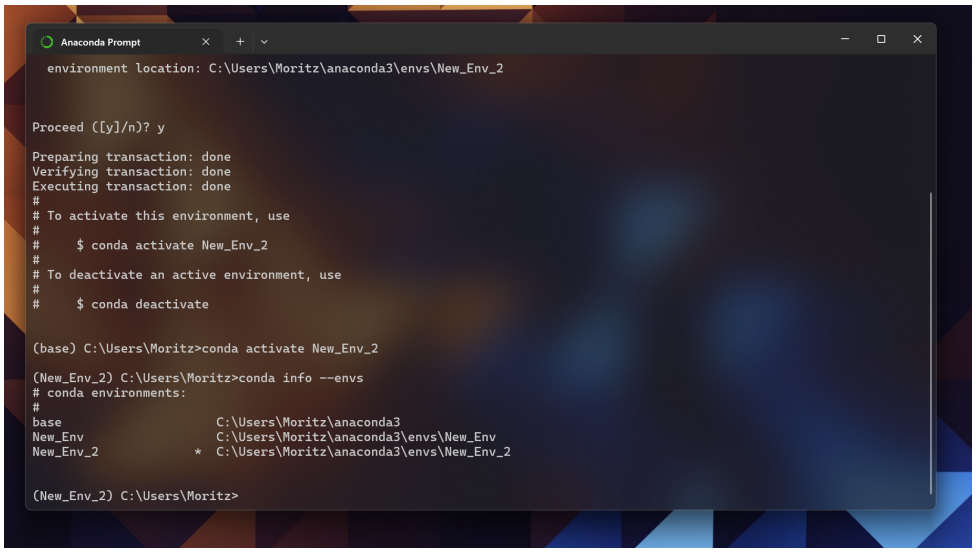

Figure 9: Activating a new environment and show existing ones

Assuming Python is not installed yet, you can do so after activating the environment, specifying the desired version:

conda install python=3.11.2

Like Navigator, you can specify the package channel from which to install packages. For example, to install numpy from the conda-forge channel:

conda install numpy -c conda-forge

To remove an installed package, you just need to specify the package name in the current environment:

conda remove numpy

Furthermore, you can also clone the environment or remove it completely:

conda create –name myclone –clone myenv

conda remove –name myenv –all

To verify installed packages or view all packages in the current environment, use:

conda list

To view a list of existing environments, use:

conda info --envs

To launch Jupyter Notebook from the command line, you'll need to install "nb_conda":

conda install nb_conda -c conda-forge

Now you can initiate Jupyter Notebook by typing "jupyter notebook" in the command line and a browser window will open.

Important: Remember, every time you open Anaconda Prompt or Terminal, you'll default to the base environment. Be sure to activate your desired environment before installing packages.

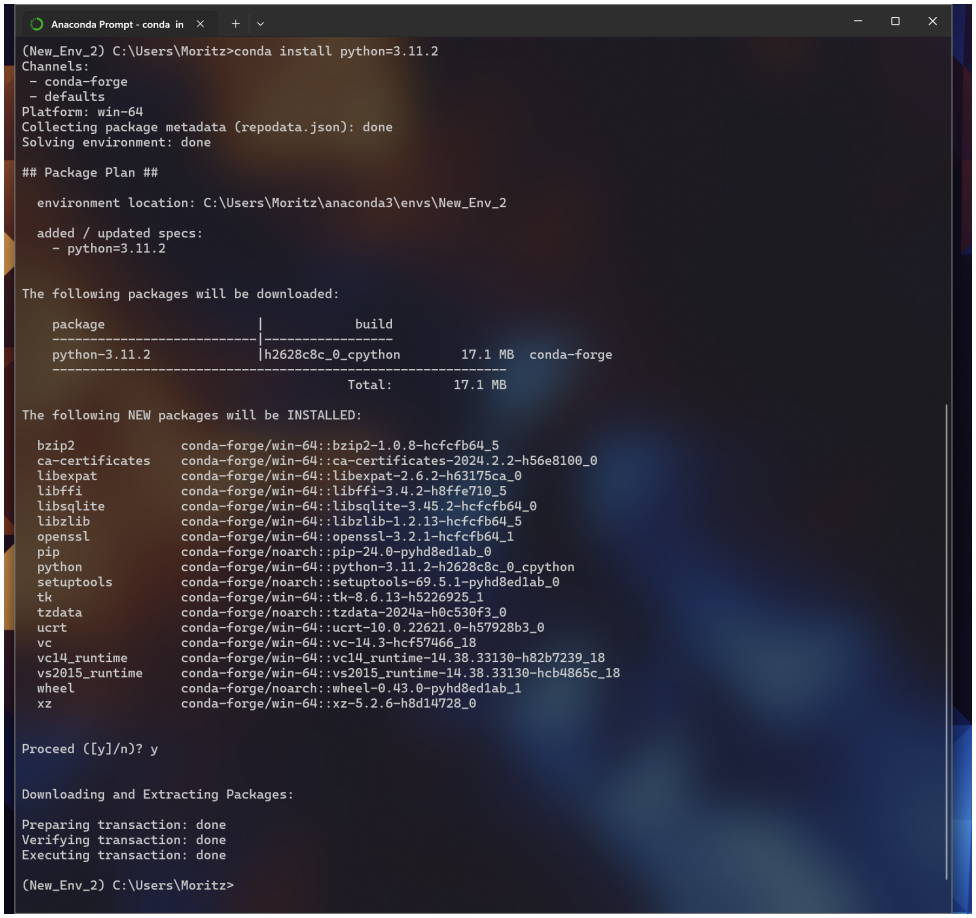

Figure 10: Installing Python with a specific version via command line

3 Jupyter Notebook

3.1 Overview

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects.

What is a “notebook”?

A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it’s a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.

3.2 Launching Jupyter Notebook

As demonstrated in the preceding sections, there are primarily two methods for launching Jupyter Notebook: using the Anaconda Navigator or employing the command line interface. Personally, I favor the command line due to its superior speed and reduced latency compared to the Navigator, although the choice is up to you. This section will provide a brief recap on how to launch Jupyter Notebook through either the Navigator or the command line. Additionally, there’s a package that warrants installation in the "base" environment beforehand to facilitate environment management within Jupyter Notebook later on.

You are already familiar with this package; it is "nb_conda". **Why do we have to install it in the base environment?**: Well, truth be told, it’s not obligatory. Alternatively, we could consistently activate our desired environment and then launch Jupyter Notebook from there. However, by installing "nb_conda" in the base environment, we can easily launch Jupyter Notebook directly from the base and seamlessly switch between various environments thereafter.

To install "nb_conda" in the base environment, you can either navigate to the Anaconda Navigator, access the "Environments" tab, locate "nb_conda" within the list of "Not installed" packages, and proceed with the installation as you’ve done previously for other environments. Alternatively, for Windows users, open the Anaconda Prompt; for macOS and Linux users, open Terminal. The base environment should be active by default. Then simply input:

conda install nb_conda -c conda-forge

After installation, there are now several ways available for launching the Jupyter Notebook.

3.2.1 Using Anaconda Navigator

In the Anaconda Navigator, head over to the "Home" tab. From there, navigate through the drop-down menu and opt for "Installed Applications". You'll find Jupyter Notebook listed, usually accompanied by a blue "Launch" button. Since "nb_conda" is now installed in the base environment, you can depart from the base environment, launch Jupyter Notebook, and subsequently switch to your preferred environment within the Notebook later. Alternatively, you can directly designate the desired environment within the Navigator's drop-down menu and proceed to launch Jupyter Notebook from there.

Should you encounter a scenario where the Jupyter Notebook application is not available from the "Home" tab, don't worry. Switch to the "Environments" tab. Here, either select the base environment or directly your desired environment, and then simply click on the green "Play" button adjacent to its name. Following this, select "Open with Jupyter Notebook" to initiate Jupyter Notebook.

3.2.2 Using the Command Line

On Windows, search for "Anaconda Prompt," while on macOS and Linux, search for "Terminal." Since we've installed "nb_conda" in the base environment, there's no need to switch environments initially. You can launch Jupyter Notebook directly from the base environment and switch to your preferred environment later. Simply type:

jupyter notebook

Alternatively, if you prefer, you can activate your desired environment first and then launch Jupyter Notebook. For instance:

conda activate myenv

jupyter notebook

3.2.3 Start Page

After initiating Jupyter Notebook, whether through the Navigator or command line, a browser window will open, displaying the directory of your designated base path. The default web page will look like Figure 11. This isn't a notebook just yet, but don't panic! There's not much to it. This is the Notebook Dashboard, specifically designed for managing your Jupyter Notebooks. Think of it as the launchpad for exploring, editing and creating your notebooks. Be aware that the dashboard will give you access only to the files and sub-folders contained within Jupyter's start-up directory (i.e., where Jupyter or Anaconda is installed). However, the start-up directory can be changed. If Jupyter Notebook is launched via the command prompt (or terminal on Unix systems), the current working directory will be the start-up directory.

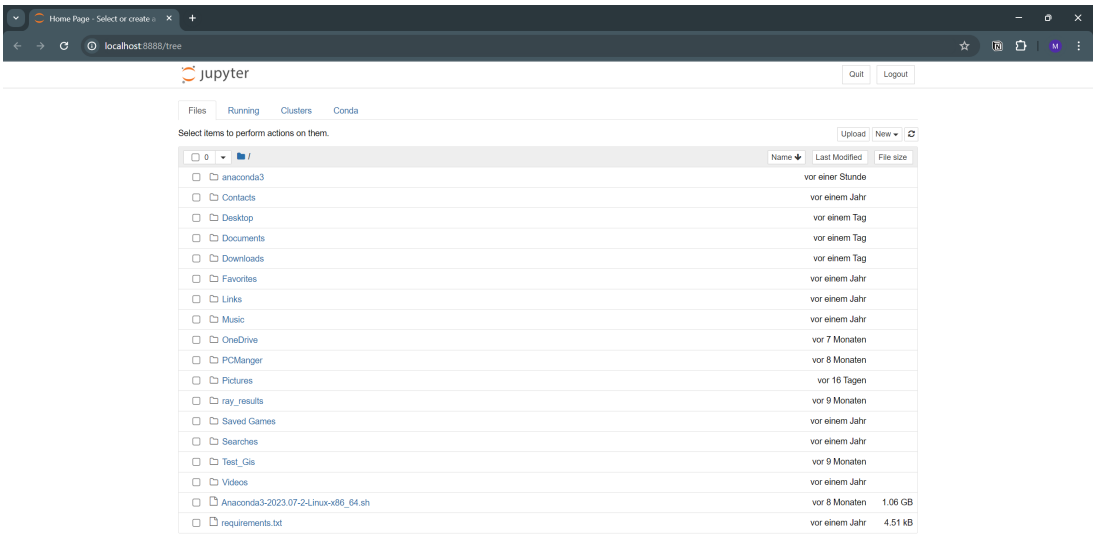

Figure 11: Start page Jupyter Notebook

3.3 Creating a new Notebook

As previously mentioned, by installing "nb_conda" in the base environment and also in each of our other environments it makes it possible to start Jupyter Notebook from any environment and then to switch between environments within the Notebook. However, first we have to create a new Notebook. For this we just click on "New" button on the top right. Now there should open the drop-down menu that should look like in Figure 12.

Here, you can select if you want to create a new notebook, text file, folder, or terminal. We want to create a new Notebook. For this, you can now select the environment that you want to use for the notebook (don't worry, you can change the environment later; it is not fixed). In my case, I have the "New_Env", "New_Env_2" and the "root" or base environment available. The current active environment is shown with a small star behind the name. Select one of the new environments that you have created before. A new browser tab will open with your new notebook (Figure 13). Each notebook will open in a new tab, because you can work in multiple notebooks simultaneously.

If you switch back to the dashboard, you will see the new file Untitled.ipynb and you should see some green text that tells you your notebook is running. Each .ipynb file is one notebook, so each time you create a new notebook, a new .ipynb file will be created.

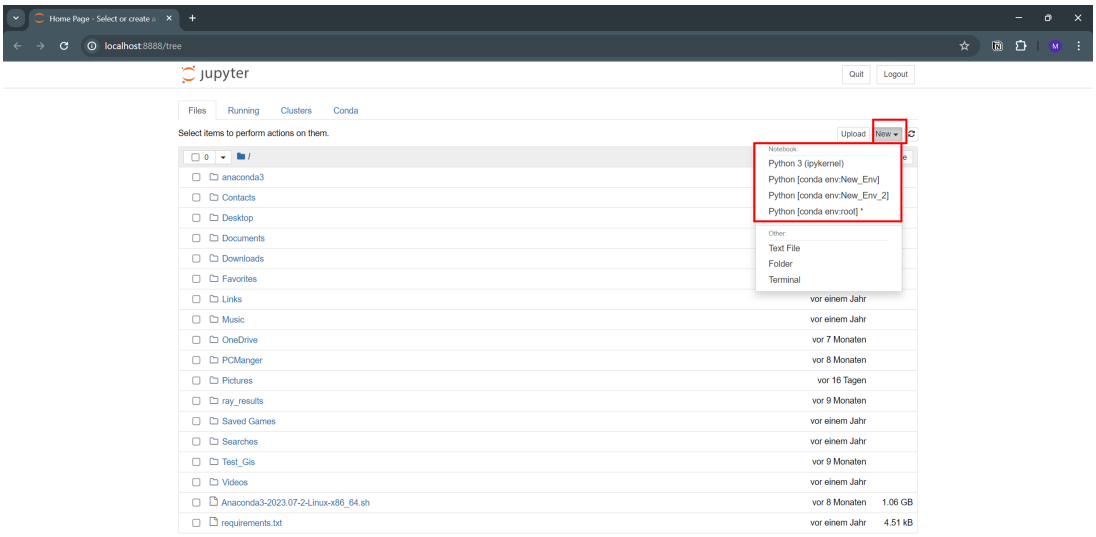

Figure 12: Selecting the environment within Jupyter Notebook

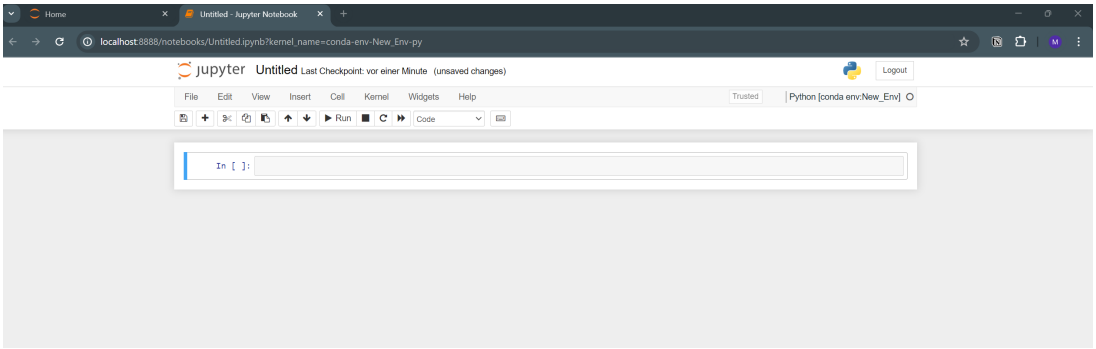

Figure 13: New notebook interface

3.4 Notebook Interface

Figure 13 shows the notebook interface. Here, you can play around and get familiar with all the menus. There are two fairly prominent terms that you should notice, which are probably new to you: cells and kernels, which are key to understanding Jupyter.

- A **kernel** is a “computational engine” that executes the code contained in a notebook document.
- A **cell** is a container for text to be displayed in the notebook or code to be executed by the notebook’s kernel.

At the top bar, click on "Kernel". In the drop-down menu, click "Change Kernel". Here, you should again see all your created environments so far, and this is also the way to

easily switch between environments in a notebook. Just select the environment you need, and the kernel will restart.

4 Summary

Now, you should feel ready to use Jupyter Notebook within the Anaconda Distribution. You should have learned:

- What is an environment, and why is it useful
- How to create a new environment using either the Navigator or the command line
- How to manage your environment including activation, deactivation, package installation, and removal
- How to install Jupyter Notebook in your environment
- How to launch Jupyter Notebook via the Navigator or the command line
- How to create a new Notebook with your desired environment
- How to change the environment within your created notebook

This tutorial does not cover the basics of Jupyter Notebook, such as running code in cells. If you're new to Jupyter, numerous beginner-friendly resources are available, including videos and tutorials. Feel free to explore those to get started smoothly. Enjoy the journey, and don't hesitate to experiment!