Virtual Environment

Anaconda

Anaconda is a distribution of Python and R for scientific computing and data science. It's widely used in the data science community because it makes it easy to manage multiple environments and packages.

1. **Install Anaconda**

   First, you'll need to download and install Anaconda. You can do this from the official Anaconda website.

   Or install Miniconda which is a minimal installer for conda. It is a small, bootstrap version of Anaconda that includes only conda, Python, the packages they depend on. (This is my preferred choice)

2. **Create a new environment**

   To create a new environment, you can use the `conda create` command, followed by the `--name` flag and the name of the environment. For instance, to create an environment named "myenv", you would run:

   ```
   conda create --name myenv
   ```

   You can also specify a Python version:

   ```
   conda create -n myenv python=3.8
   ```

3. **Activate the environment**

   Before you start using the environment, you'll need to activate it. You can do this with the `conda activate` command:

   ```
   conda activate myenv
   ```

4. **Install packages**

Once you're in the environment, you can install packages using the `conda install` command. For example, to install jupyter notebooks, you would run:

```
conda install jupyter
```

5. **Deactivate the environment**

   When you're done working in the environment, you can deactivate it with the `conda deactivate` command.

Pyenv

Pyenv is a tool that allows you to install multiple versions of Python and easily switch between them.

1. **Install Pyenv**

   The installation process will depend on your operating system. For most Unix-like systems, you can use the Pyenv installer script. You can find detailed instructions on the official Pyenv Github page.

2. **Install a Python version**

   To install a new version of Python, you can use the `pyenv install` command. For instance, to install Python 3.9.0, you would run:

   ```
   pyenv install 3.9.0
   ```

3. **Create a virtual environment**

   Pyenv includes a plugin called `pyenv-virtualenv` which allows you to manage virtual environments. First, install the plugin following the instructions on the official Github page.

   After installing `pyenv-virtualenv`, you can create a new virtual environment with the `pyenv virtualenv` command. This command takes two arguments: the Python version you want to use, and the name of the environment. For example, to create an environment named "myenv" with Python 3.9.0, you would run:

   ```
   pyenv virtualenv 3.9.0 myenv
   ```

4. **Activate the environment**

   To activate the environment, you can use the `pyenv activate` command:

   ```
   pyenv activate myenv
   ```

5. **Deactivate the environment**

   When you're done working in the environment, you can deactivate it with the `pyenv deactivate` command.

By using virtual environments, you can ensure that each of your Python projects has its own isolated space to run and store dependencies, preventing conflicts between different versions of packages or Python itself.