setup_venv

December 28, 2025

1 Python Virtual Environment

A Python virtual environment (venv) is an isolated environment that allows you to run a specific Python version with its own set of packages, independent of the global Python installation.

Think of it as a sandbox for your project: it keeps your dependencies separate from other projects or the system Python.

2 Why Use a Virtual Environment?

1. **Avoid Package Conflicts**
   Different projects may require different versions of the same package.
   Example:
   - Project A needs `Django 4.2`
   - Project B needs `Django 3.2`
2. **Project Isolation**
   Each project can maintain its own dependencies.
   Deleting a virtual environment does not affect other projects or system Python.
3. **Cleaner Deployment**
   Using `requirements.txt`, you can recreate the exact environment on another machine.
4. **Testing Different Python Versions**
   You can create virtual environments with specific Python versions if multiple are installed.

---

3 Creating a Virtual Environment (`venv`) - Step by Step

3.1 1. Check Python Version

~~~
Make sure Python 3.3+ is installed:

```python
python3 --version
~~~

3.2 2. Create a Virtual Environment

Create a new virtual environment called myenv.
Run the following command in your project directory: <b>python3 -m venv myenv</b>
myenv is the name of your virtual environment folder.
A folder named myenv will be created containing an isolated Python environment.

3.3 3. Activate the Virtual Environment

- Linux/macOS:

```
source myenv/bin/activate
```

- Windows (Command Prompt):

```
myenv\Scripts\activate
```

- Windows (PowerShell):

```
myenv\Scripts\Activate.ps1
```

After activation, the terminal prompt shows (myenv) at the beginning.

3.4 4. Install Packages

Once the environment is active, install packages locally:

```
pip install package_name
```

Example:

```
pip install numpy pandas matplotlib
```

3.5 5. Deactivate the Environment

To exit the virtual environment:

```
deactivate
```