Virtual Environment in Python and Its Uses

🔹 What is a Virtual Environment?

A **virtual environment** is an isolated workspace on your system where you can install Python packages specific to a project, without affecting the global Python installation.

🔹 Why Use a Virtual Environment?

Reason	Explanation
Project Isolation	Avoid conflicts between dependencies of different projects.
Version Control	Maintain specific package versions for each project.
Clean Setup	Keep global Python clean from unnecessary packages.
Reproducibility	Easier for teams to replicate the same setup using `requirements.txt`.

🔹 How to Create and Use a Virtual Environment

✅Step 1: Install `virtualenv` (if not already installed)

```
pip install virtualenv
```

✅Step 2: Create a Virtual Environment

```
virtualenv venv
```

This will create a folder named `venv` containing a new Python environment.

✅Step 3: Activate the Virtual Environment

- **Windows:**

```
venv\Scripts\activate
```

- **Linux/macOS:**

```
source venv/bin/activate
```

You will now see `(venv)` at the beginning of your terminal, indicating it's active.

✅Step 4: Install Packages

```
pip install django djangorestframework
```

Packages are now installed **only** inside `venv`.

✅Step 5: Freeze Requirements (Optional but Recommended)

```
pip freeze > requirements.txt
```

This creates a list of all installed packages and versions.

✅Step 6: Deactivate the Environment

```
deactivate
```

Switches back to the global Python environment.

⬜ Common Commands

Command	Description
`virtualenv venv`	Create a virtual environment
`source venv/bin/activate`	Activate environment on Linux/macOS
`venv\Scripts\activate`	Activate environment on Windows
`deactivate`	Exit the virtual environment
`pip freeze > requirements.txt`	Save all installed packages to a file
`pip install -r requirements.txt`	Install packages from a saved requirements file

⚠️Summary

- A virtual environment keeps your project dependencies isolated.
- It avoids version conflicts between multiple projects.
- It's considered a **best practice** for every Python or Django project.