Virtual Environment in Python

1️⃣ Importance of a Virtual Environment

- A **virtual environment** is an **isolated Python environment** that allows you to manage dependencies for a specific project without affecting the global Python installation.
- **Why it's important:**
    1. **Dependency Isolation:** Different projects may require different versions of packages (e.g., Django 4.2 vs Django 3.2). Virtual environments prevent conflicts.
    2. **Project Organization:** Keeps project-specific packages separate from global packages.
    3. **Safe Testing:** Allows testing new packages or versions without risking system-wide Python setup.
    4. **Reproducibility:** Helps share your project with others using a requirements.txt file listing exact packages.

2️⃣ Creating and Using Virtual Environments

Using venv (built-in in Python 3.3+)

Step 1: Create a virtual environment

python -m venv myenv

- myenv is the name of the virtual environment folder.

Step 2: Activate the virtual environment

- **Windows:**

myenv\Scripts\activate

- **Linux/macOS:**

source myenv/bin/activate

- After activation, your terminal will show the environment name, e.g., (myenv).

Step 3: Install packages inside the virtual environment

pip install django

Step 4: Deactivate when done

deactivate

Using virtualenv (alternative tool)

- Install virtualenv (if not already installed):

pip install virtualenv

- Create a virtual environment:

virtualenv myenv

- Activate and deactivate the same way as venv.

3️⃣ Best Practices

1. Always use a **virtual environment for each Python project**.
2. Keep a requirements.txt file:

pip freeze > requirements.txt

- Others can install the same dependencies using:

pip install -r requirements.txt

3. Avoid installing project packages globally to prevent conflicts.