**Virtual Environments in Python (Production Notes)**

In production, developers usually **do not install packages globally**. Instead, they use **virtual environments** to keep project dependencies isolated.

Two popular approaches are:

1. **Built-in venv module**
2. **Using uv (modern tool, much faster)**

**🔹 1. Virtual Environment using venv**

**Steps:**

1. **Make a project folder**
2. mkdir myproject
3. cd myproject
4. **Create a virtual environment**
5. python -m venv venv
6. **Activate the virtual environment**
   * **Windows (PowerShell / CMD):**
   * venv\Scripts\activate
   * **macOS/Linux (bash/zsh):**
   * source venv/bin/activate

✅ You will see (venv) in your terminal, meaning it’s active.

1. **Deactivate**
2. deactivate

**Package Management in venv**

* **Upgrade pip**
* python -m pip install --upgrade pip
* **Install packages**
* pip install requests
* **Freeze dependencies (save to file)**
* pip freeze > requirements.txt
* **Install from requirements.txt**
* pip install -r requirements.txt

👉 This is very common in production for sharing project dependencies.

**🔹 2. Virtual Environment using uv**

[uv](https://github.com/astral-sh/uv?utm_source=chatgpt.com) is a **new, faster package and virtual environment manager** (alternative to pip + venv). It’s written in Rust, very efficient, and gaining popularity.

**Steps:**

1. **Install uv (once)**
2. pip install uv
3. **Create a virtual environment**
4. uv venv
5. **Activate**
   * **Windows:**
   * .venv\Scripts\activate
   * **macOS/Linux:**
   * source .venv/bin/activate
6. **Install packages with uv**
7. uv pip install requests
8. **Export dependencies**
9. uv pip freeze > requirements.txt
10. **Install from requirements.txt**
11. uv pip install -r requirements.txt

**✅ Key Differences: venv vs uv**

| **Feature** | **venv + pip (traditional)** | **uv (modern)** |
| --- | --- | --- |
| Speed of install | Slower (downloads each package) | Much faster (caching + parallel installs) |
| Built-in? | Yes (comes with Python) | No (needs installation) |
| Ease of use | Simple, widely used | Modern, powerful, improving fast |
| Dependency handling | Manual (pip freeze) | Cleaner & faster |

📌 **In production**:

* Most companies still use venv + pip because it’s standard and built-in.
* But **uv** is becoming popular for **fast package management**.