1. What is `pip`?

`pip` is the **standard package manager** for Python. It allows you to **install, upgrade, and remove** external Python libraries (also called packages or modules) from PyPI (Python Package Index).

Common pip Commands

Command	Description
`pip install package-name`	Install a package
`pip uninstall package-name`	Uninstall a package
`pip list`	List installed packages
`pip show package-name`	Show package details
`pip install -r requirements.txt`	Install from a requirements file
`pip install package==1.2.3`	Install a specific version
`pip freeze`	Output installed packages + versions
`pip freeze > requirements.txt`	Save current environment to a file

Example:

```
pip install requests
```

Then use it:

```
import requests
response = requests.get("https://example.com")
print(response.status_code)
```

2. What is a Virtual Environment?

A **virtual environment** is an isolated Python environment where you can install packages separately without affecting the global Python installation to run and test your Python projects.

This helps:

- Avoid version conflicts between projects.
- Keep your projects clean and portable.

Think of a virtual environment as a separate container for each Python project. Each container:

- Has its own Python interpreter
- Has its own set of installed packages
- Is isolated from other virtual environments
- Can have different versions of the same package

Why Use Virtual Environments?
Let's say Project A uses Django 3.2, and Project B uses Django 4.0. Without a virtual environment, installing both versions would cause conflicts. With virtual environments:

- Each project has its own dependencies.
- No interference between projects.

How to Use Virtual Environments

Step 1: Install virtualenv (optional)

- pip install virtualenv

On Python 3.3+, you can use the built-in venv module.

Step 2: Create a Virtual Environment

- python -m venv env

This creates a folder env/ with the isolated environment.

Step 3: Activate the Environment

OS	Command
Windows	env\Scripts\activate
macOS/Linux	source env/bin/activate

Once activated, you'll see (env) at the beginning of your terminal and your shell prompt change to something like:

(env) C:\Users\Raj\project>

Step 4: Install Packages Inside Environment

- (env) C:\Users\Raj\project> pip install flask

This installs Flask only for this environment.

Step 5: Freeze Dependencies

To save all installed packages:

- pip freeze > requirements.txt

Step 6: Deactivate the Environment

- (env) C:\Users\Raj\project>deactivate

This returns you to your system's global Python.

Summary: pip vs venv

Tool	Purpose
pip	Installs Python packages
venv	Creates isolated environments
requirements.txt	Stores package list to recreate env

Mini Assignment

1. Create a virtual environment named myenv
2. Activate it
3. Install the requests and flask packages
4. Save the installed packages into requirements.txt
5. Deactivate the environment