



Introduction

- A Python **environment** is a self-contained directory with its own installation of Python and libraries.
- Helps avoid conflicts between projects.
- Types of environments:
 - Virtual Environments (venv, virtualenv) and
 - Conda Environments.

Difference Between venv, virtualenv, and Conda

venv:

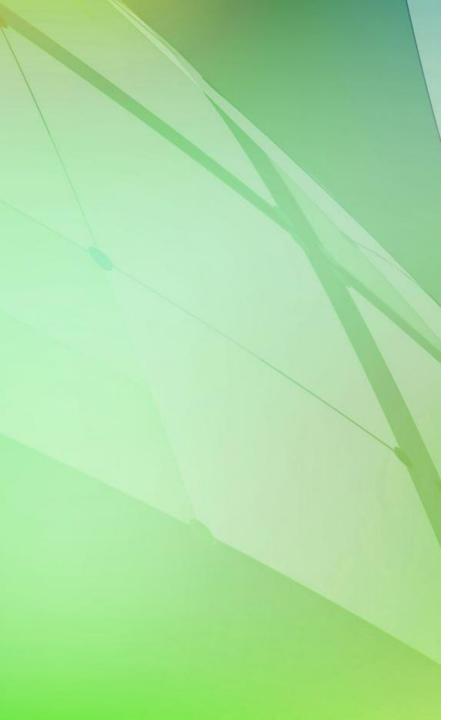
- Comes built-in with Python
- Lightweight and simple, best for standard projects.
- Requires pip for package management.

virtualenv:

- Third-party package that extends venv.
- Uses pip for package management.

Conda:

- Part of the Anaconda distribution. Supports package management for Python nd non-Python dependencies.
- Uses conda for package management but can also install packages via pip if needed.



Virtual Environments with venv

- Creating a Virtual Environment:
- python -m venv myenv
- Activating the Environment:
 - Windows: myenv\Scripts\activate
 - Mac/Linux: source myenv/bin/activate
 - Installing Packages: pip install package_name
 - Deactivating the Environment: deactivate



Virtual Environments with virtualenv

Installation:

• pip install virtualenv

Creating a Virtual Environment:

virtualenv myenv

Activation and Deactivation:

• (Same as venv)

Managing Dependencies with Requirements File

Create a requirements file:

• pip freeze > requirements.txt

Install dependencies from a requirements file:

• pip install -r requirements.txt

Conda Environments

- Creating a Conda Environment:
 - conda create --name myenv python=3.9
- Activating the Environment:
 - conda activate myenv
- Installing Packages:
 - conda install package_name
- Exporting and Importing Environments:
 - Export:conda env export > environment.yml
 - Import:conda env create -f environment.yml

You are now a Master in Environments!

End