What is MVT in Django? How does it work :

In Django, MVT stands for Model-View-Template, an architectural pattern for building web applications. It is Django's variation of the widely used Model-View-Controller (MVC) pattern,

Components of Django MVT:

In MVT, there are three components :

1. **Model:** This is the data access layer, a Python class that defines the structure and behavior of your application's data. Each model typically maps to a single database table.

2. **View:** The view is the logic layer, a Python function or class that receives web requests, interacts with the relevant model to fetch or update data, and decides what data to send to the template.

3. **Template**: This is the presentation layer, typically an HTML file with embedded Django Template Language (DTL) syntax (using `{{ variables }}` and `{% tags %}`) to render dynamic content. It defines the structure and layout of the web page presented to the user

What is Python ENV (Python environment) :

A virtual environment is an isolated Python environment that allows you to manage dependencies for each project separately.

Why do we need a Virtual Environment :

1. Avoid Dependency Conflicts: Different projects may require different versions of the same library

2. Isolate Project Environments: Keeps each project’s packages and settings separate from others and from the system Python.

3. Simplifies Project Management: Makes it easier to manage and replicate project-specific setups.

4. Prevents System Interference: Avoids accidentally modifying or breaking the global Python environment.

5. Enables Reproducibility: Ensures consistent behavior across development, testing and deployment environments.

When and Where to Use a Virtual Environment :

A virtual environment should be used for every Python project. By default, all Python projects share the same location to store packages. It prevents conflicts between projects and avoids affecting the system-wide Python installation. Tools like venv or virtualenv are commonly used to create them.

How to Create a Virtual Environment in Python

Create a virtual environment (named `venv` in this example)
bush : python -m venv venv
Activate the virtual environment (Windows)
bush: venv\Scripts\activate
Activate the virtual environment (Linux)
bush : source venv/bin/activate

Deactivate Python Virtual Environment

Once you are done with the work, you can deactivate the virtual environment with the following command:
(virtualenv_name)$ deactivate

How to install Django :

Prerequisites

Before installing Django, you must have Python installed on your system. Python 3.4 and later versions include `pip` by default

1. Create and Activate a Virtual Environment

It is highly recommended to use a virtual environment for every Django project.

After activating the Virtual Environment

1. Install Django

With your virtual environment activated, install Django using pip:

pip install Django

2. Verify the Installation

You can check if Django was installed successfully and view the version number

python -m django --version

4. Create a Django Project and Run the Server

Once installed, you can start using Django commands to build your application:

Create a new project:

Bush

django-admin startproject mysite.

Run the development server:

python manage.py runserver