# **INTRODUCTION TO DJANGO**

* **What is a Web Framework ?**

A web development framework is a set of resources and tools for software developers to build and manage [web applications](https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app), [web services](https://www.techtarget.com/searchapparchitecture/definition/Web-services) and websites, as well as to develop application programming interfaces ([APIs](https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API)). Web development frameworks are also referred to as web application frameworks or simply web frameworks.

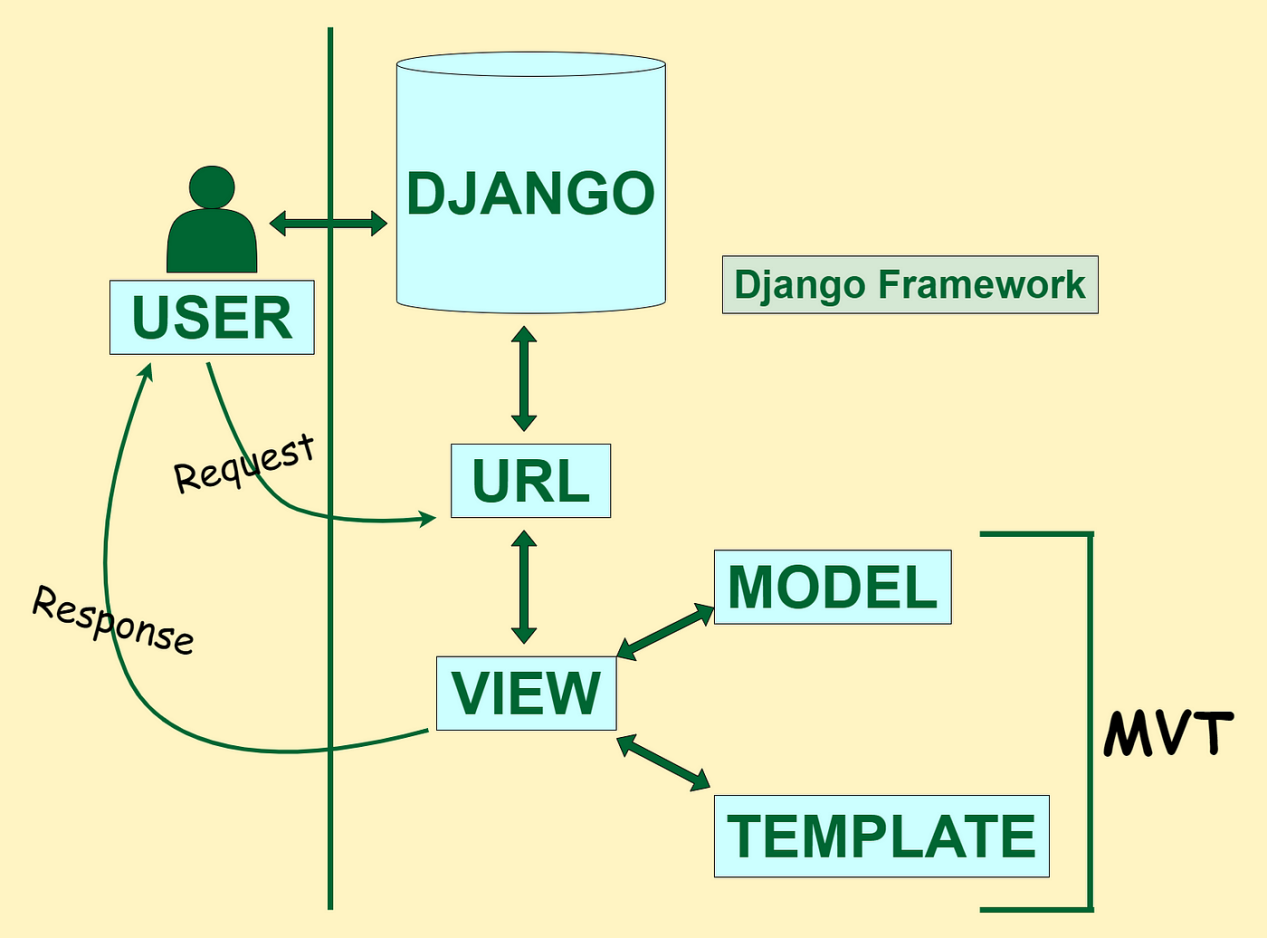
Web development frameworks enable developers to build applications that can run on well-known technology [stacks](https://www.techtarget.com/searchapparchitecture/definition/software-stack) such as the [Linux](https://www.techtarget.com/searchdatacenter/definition/Linux-operating-system), [Apache](https://www.techtarget.com/whatis/definition/Apache), [MySQL](https://www.techtarget.com/searchoracle/definition/MySQL) and [PHP](https://www.techtarget.com/whatis/definition/PHP-Hypertext-Preprocessor) ([LAMP](https://www.techtarget.com/whatis/definition/LAMP-Linux-Apache-MySQL-PHP)) stack. Most frameworks provide a wide range of features and functionality that help streamline application development. For example, they might include any of the following components:

1. Application [templates](https://www.techtarget.com/whatis/definition/template) for presenting information within a browser.
2. Programming environment for scripting the flow of information.
3. APIs for accessing back-end data resources.
4. [Code](https://www.techtarget.com/whatis/definition/code) libraries with pre-built components and code snippets.
5. Support for [debugging](https://www.techtarget.com/searchsoftwarequality/definition/debugging), quality assurance ([QA](https://www.techtarget.com/searchsoftwarequality/definition/quality-assurance)) testing and code reusability.

* **What is DJANGO ?**

Django is a free, open-source, high level **python web framework** that follows the **Model-View-Template (MVT)** architectural pattern. It is used to build and maintain websites and web applications. It is known for its **speed**, **security** and **scalability**.

* **About MVT pattern**



1. Here, a user ****requests**** for a resource to the Django, Django works as a controller and check to the available resource in URL.
2. If URL maps, ****a view is called**** that interact with model and template, it renders a template.
3. Django responds back to the user and sends a template as a ****response****.

## What is a Virtual Environment?

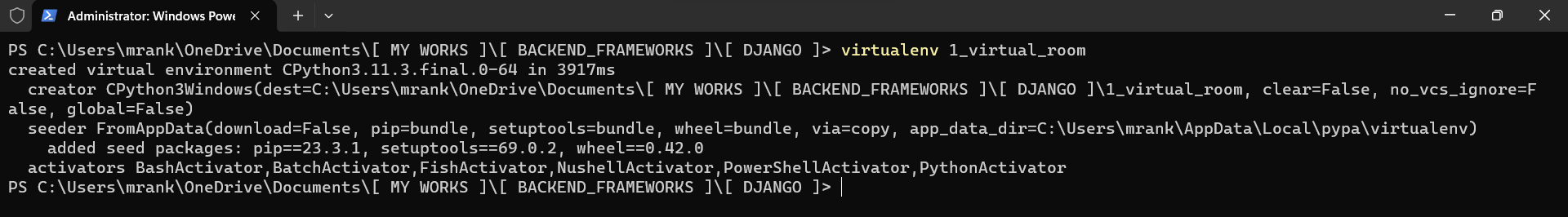
A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated [Python](https://www.geeksforgeeks.org/python-programming-language/) virtual environments for them. This is one of the most important tools that most Python developers use.

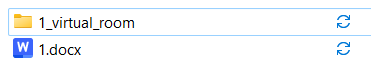
Imagine a scenario where you are working on two web-based Python projects one of them uses [Django](https://www.geeksforgeeks.org/django-tutorial/) 4.0 and the other uses Django 4.1 (check for the latest Django versions and so on). In such situations, we need to create virtual environment in Python that can be really useful to maintain the dependencies of both projects.

* **virtualenv**

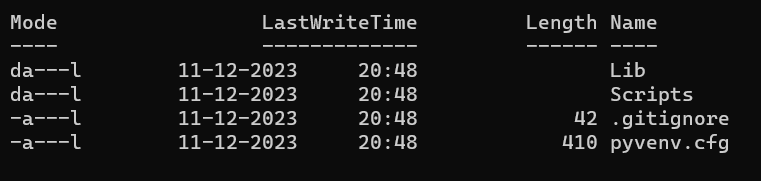
1. pip install virtualenv
2. We use a module named **virtualenv** which is a tool to create isolated Python environments. virtualenv creates a folder that contains all the necessary executables to use the packages that a Python project would need.
3. <https://pypi.org/project/virtualenv/>
4. Creating a different virtual room/environment

“virtualenv <virtual\_env\_name>”





1. Activating the virtual environment
2. “cd <environment\_name”
3. Files that are by default present in <virtual\_environment>

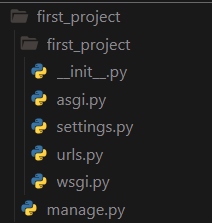


1. Activate virtual environment: “Scripts\activate”
2. Virtual environment gets started (you will see a parenthesis prompt at the start of each line)
3. Now cd to <virtual\_environment> folder
4. Open vs code by writing “code .”
5. Create your first project in Django

“django-admin startproject <project\_name>”

* **INSIGHT ON DJANGO**

Project name : - “first\_project”



**“ \_\_init\_\_.py “**

1. It is a special file that is used to mark a directory as a python package.
2. When a python package is imported, the code in **\_\_init\_\_.py** file is executed.

1. This can be used to initialize package such as loading data or setting up logging.

D) It is also use to define the app’s configuration such as app’s name, label and URL patterns.

**“ asgi.py “**

1. It stands for **Asynchronous Server Gateway Interface.**
2. It is an entry point for ASGI compatible web servers to serve your project.
3. It is generated by startproject command and is located in the root of the Django project.
4. It contains a single function name **“application()”**, which is responsible for handling requests from the web server.
5. This file simply imports the **get\_asgi\_application()** function from the **django.core.asgi** module and assigns it to a variable.
6. The **get\_asgi\_application()** function returns an **ASGI application object** that can be used by the web server to handle incoming requests.

**“ settings.py ”**

1. It s a core file in Django projects and located in the project’s top-level directory .
2. It contains configuration information for the Django project.
3. It is important to note that the settings.py should not be version controlled as this can lead to security problems. Instead each environment that the project is deployed to should have its own settings.py which is specific to that environment.
4. Here are some of the important settings that are typically defined in the settings.py file:

**DEBUG**: This setting controls whether or not Django will display debug messages.

**ALLOWED\_HOSTS**: This setting controls which hosts are allowed to access the Django project.

**MIDDLEWARE**: This setting controls which middleware classes are used by the Django project.

**ROOT\_URLCONF**: This setting controls the URL configuration for the Django project.

**INSTALLED\_APPS**: This setting controls which applications are installed in the Django project.

**DATABASES**: This setting controls the database configuration for the Django project.

**TEMPLATES**: This setting controls the template configuration for the Django project.

**STATIC\_FILES**: This setting controls the static file configuration for the Django project.