**What is Framework?**

* A framework is a structure that you can build software on. The purpose of the framework is to allow developer to focus on building a unique feature of their project rather than writing code from scratch.
* Framework is a set of (predefined class or function) tools and modules that can be reused for various projects.
* Framework is a set of conceptual structure and guideline, used to build something useful.
* Example :- Brick

**Why use Framework?**

* Collection of tools
* No need to write a code from scratch
* Save time
* Improve productivity
* Clean code
* Reusable code
* Easy debugging
* Testing

**Web Framework?**

A web framework (WF) or web application framework (WAF) is a software framework that is designed to support the development of web applications including web services, web resources, and web APIs.

**What is Virtual Environment?**

A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them.

Steps:

1.pip install virtualenv

2.python -m virtualenv env\_name or virtualenv env\_name

3.env\_name\scripts\activate

4.pip install Django == 3.0

5.django-admin --version or django -m --version

6.deactivate

**What is Django?**

* Django is a back-end server side web framework.
* Django makes it easier to build web pages using Python.
* Django is a free and open-source web framework.
* Django is a **Python-based web framework** that allows you to quickly create efficient web applications.
* It follows the **Model-View-Template** (MVT) architectural pattern.
* It is maintained by **the Django Software Foundation** (DSF)
* Django was created in 2003 as an internal project at Lowrence Journal-World News Paper for their web development.
* The Original authors of Django Framework are: Adrian Holovaty, Simon Willison
* After Testing this framework with heavy traffics, Developers released for the public as open source framework on July 21st 2005.
* It is also called **batteries** included framework because Django provides built-in features for everything including Django Admin Interface, default database – SQLlite3, etc.
* Django takes care of much of the development, so you can focus on writing your app without needing to reinvent the wheel.

**Application**

Instagram, Mozilla, Disqus, Pinterest, etc.

**Advantages:**

* **Rapid Development:** Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly. Fully loaded: Django includes dozens of extras we can use to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks.
* **Security:** Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.
* **Scalable:** Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.
* **Fully loaded:** Django includes various helping task modules and libraries which can be used to handle common Web development tasks.
* **Versatile:** Django is versatile in nature which allows it to build applications for different-different domains. Now a days, Companies are using Django to build various types of applications like: content management systems, social networks sites or scientific computing platforms etc.
* **Open Source:** Django is an open source web application framework. It is publicly available without cost.

**Disadvantage:**

**1. Mastering**

Developers who are migrating from other programming languages to Python with Django often find it difficult to learn it. The framework has a lot of configurations and settings that need to be configured out by Developers. Even though it is a simple and easy-to-understand framework it is often claimed as hard to master.

**2. Not suitable for small Projects**

Django has a lot of code that requires Server Computation Power and can take a lot of time, which takes a toll on small applications. Same reason why we wouldn’t use a drilling machine to punch a staple hole on paper. You can opt for other frameworks such as Flask to handle smaller projects.

**3. Monolithic**

Django does not provide a lot of freedom when it comes to project architecture, Django has its own way of doing things. If a developer tries to implement something out of the file structure then there is no way Django can access the file. Django is very specific **when it comes to performing tasks in a certain way.**

**4. Tightly Coupled Components**

Usually, frameworks maintain their components loose so that developers have an easy time moving them. Django’s Components are tightly coupled due to which developers have to focus on installing all the components at the same time.

Many developers find it annoying and time-consuming while migrating their projects around systems.

**5. Cannot Handle Multiple Requests**

The majority of the web frameworks improve the execution of web applications by offering support to handle multiple requests at once.But Django, being a modern web framework, does not support handling multiple requests simultaneously. Therefore developers have to come up with different approaches to deal with multiple requests rapidly.

**MVT Architecture:**

* MVT architecture is the software design pattern used by the Django web framework.
* MVT stands for Model – View – Template.

1)Model

The model is going to act as the interface of your data. It is responsible for maintaining data in database.

2)Template

Templates are responsible for the entire User Interface completely. It handles all the static parts of the webpage along with the HTML, which the users visiting the webpage will perceive.

3) Views

In Django, Views act as a link between the Model data and the Templates.

A request handler that returns the relevant template and content - based on the request from the user.

Note: Just like the controller in MVC, views in Django MVT are responsible for handling all the business logic behind the web app. It acts as a bridge between Models and Templates.

Model --> Table in DataBase/Class in Python -->models.py

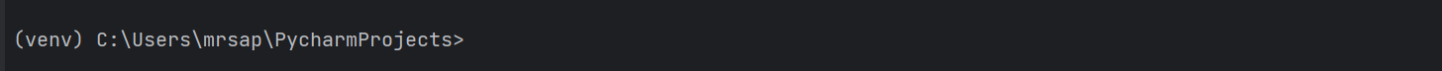
View --> Business Logic --> FBVs/CBVs -->views.py

Template --> Presentation --> HTML file -->templ.html

**How to install django:**

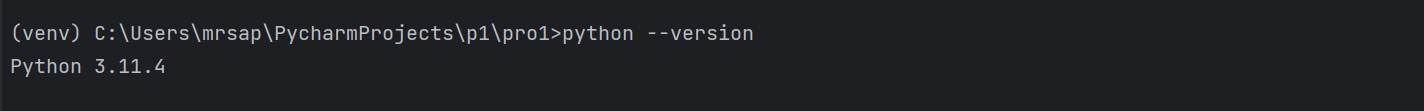
**Step 1:** Open pycharm terminal

**Step 2:** Check virtual environment is activated or not.



**Step 3**: Make sure Python is already installed in our system

python --version



**Step 4:** Install django by using pip

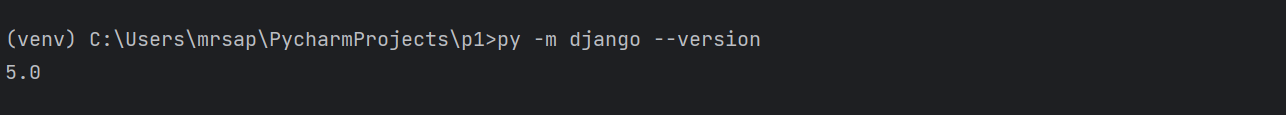
pip install django or pip install Django==version

A screen shot of a computer program

Description automatically generated

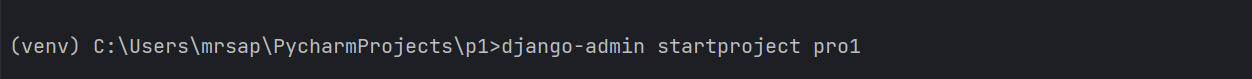
**Step 5:** To check django version:

py -m django –version



**Step 6:** Create Django Project

django-admin startproject <project\_name>



**Step 7:** Change Directory

cd <project\_name>



**Step 8:** Run development server

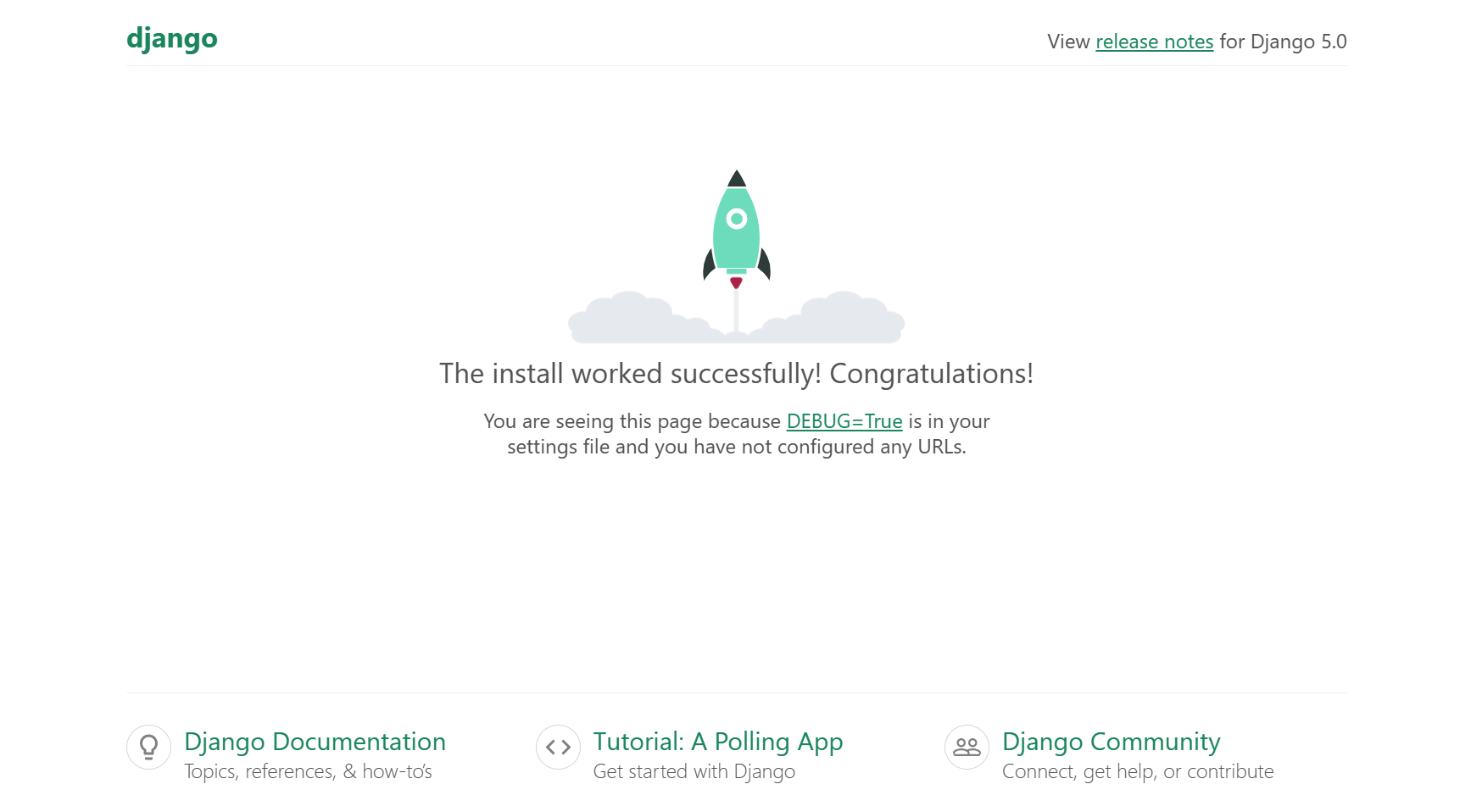
py manage.py runserver

python manage.py runserver

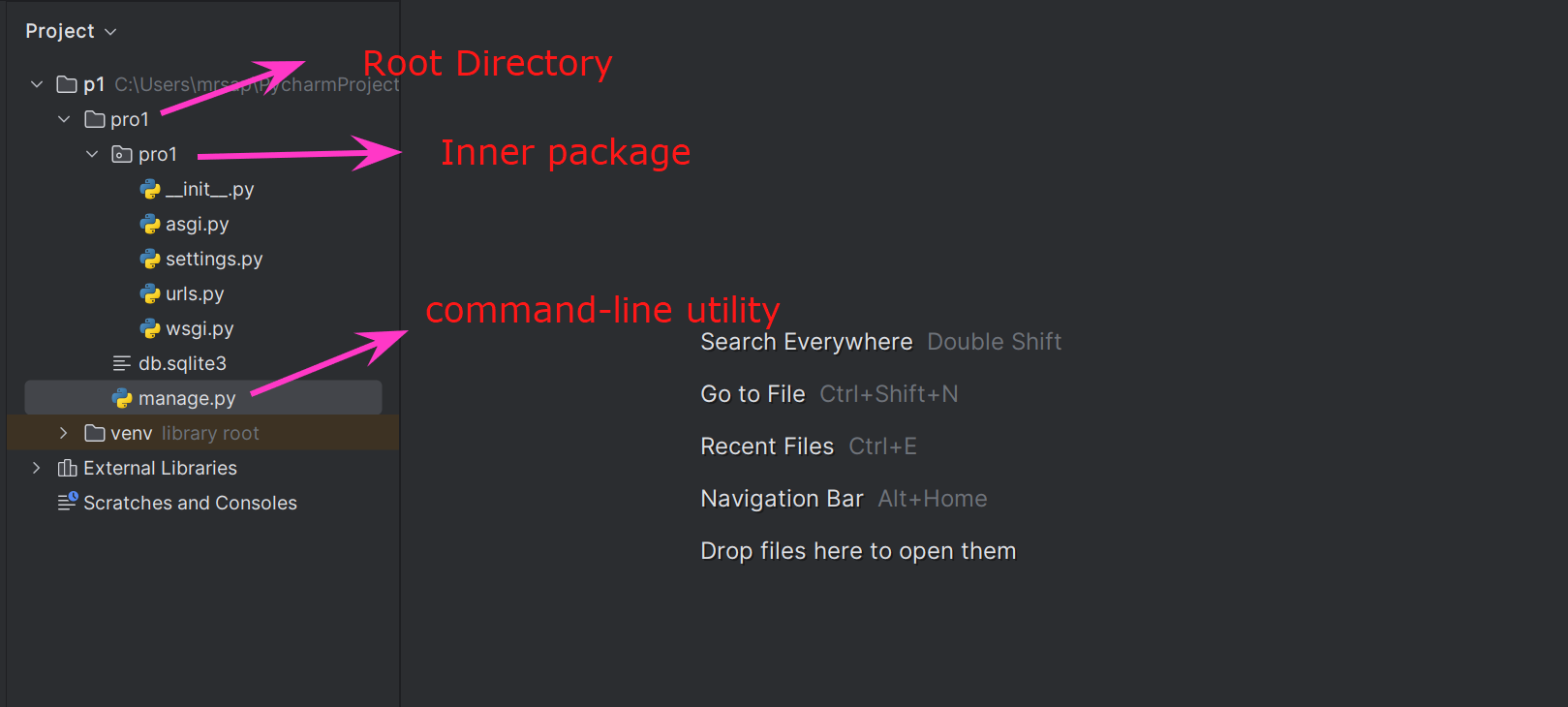
A black screen with white text

Description automatically generated

**Step 9:** Click on development server link



**Folder Structure:**



**\_\_init\_\_.py:**

It is a blank python script. Because of this special file name, Django treated this folder as python package.

Note: If any folder contains \_\_init\_\_.py file then only that folder is treated as Python

package. But this rule is applicable until Python 3.3 Version.

**settings.py:**

In this file we have to specify all our project settings and configurations like

installed applications, middleware configurations, database configurations, etc.

**urls.py:**

* In urls.py file we have to store all our URL patterns of our project.
* For every view (web page), we have to define separate URL pattern. End user can use URL patterns to access our webpages.

**wsgi.py/asgi.py:**

* The full form of wsgi is Web Server Gateway Interface and asgi is asynchronous Server Gateway Interface.
* It describes the way how servers interact with the applications.
* ASGI is backward-compatible with WSGI and supports multiple servers and application frameworks.
* We can use this file while deploying our application in production on online server.

**manage.py:**

* The most commonly used python script is manage.py.
* It is a command line utility to interact with Django project in various ways like to run development server, run tests, create migrations etc.
* We will use this file for debugging, deploying, and running our web applications.