1. What is a framework?

When you work on software, you might use any framework to enhance the quality of your application. **Frameworks** provide many advantages to the developers and reduce the time and effort required in the development process. They enable developers to write clean code that is easily understandable by others. It serves as a foundation.

Writing code is complex. And writing code that others can understand and manage is even more difficult because you must deal with syntax, declarations, performance, staying consistent, and other issues.

Software frameworks provide a template in which almost all general tasks have been handled. This allows you to focus on the core software development rather than the details of the process. Given that there is a set structure, it is simple to collaborate with others.

It's a good idea to use a software framework rather than re-inventing the wheel from scratch for numerous reasons. And perhaps the most important reason is that you won't have to write everything from scratch. This reduces the possibility of adding errors to your code.

* It helps you avoid duplicate and redundant code.
* It makes it easier for developers who did not write the code to test and debug it.
* Frameworks are maintained by a group of people who test them so that you can use them with confidence.
* They help you write clean and secure code.
* The time required to develop an application is significantly reduced because you can now focus on writing project-specific code.

1. What is Django, what are the advantages of using Django?

Django is an MVT web framework used to build web applications. It is a web framework, with robustness and simplicity to help web developers write clean, efficient and powerful code. It is among the most famous web frameworks out there in the world and it’s one of the most used frameworks as well. It’s used by Instagram, Youtube, Google and even NASA for their website.

Django follows the MVT design pattern (Model View Template).

* Model - The data you want to present, usually data from a database.
* View - A request handler that returns the relevant template and content - based on the request from the user.
* Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.

Advantages of Django

1.Batteries includedDjango prides itself as a batteries-included framework. What that means is that it comes with a lot of stuff out of the box, that you may or may not use depending on your application. Instead of having to write your own code (the power), you just need to import the packages that you want to use.

2. PythonSince Django uses Python, it leverages some of the fame and power of python to its own benefit. Python is arguably one of the easiest -if not the easiest- programming language to learn for beginners, and it’s also quite popular in introductory computer science courses around the world. The 2017 Stackoverflow Developers Survey revealed that Python is now more common than PHP and Python jobs pay better than C# and C++.3. CommunityDjango’s community is one of the best things about it, they are helpful and actively working on making the framework more beginner-friendly and stabilizing the framework while adding new features. Django’s documentation is quite thorough and is useful as a standalone tutorial, it will help you wrap your head around various features so you can use it as a primary source of information.

4. ScalableMost developers, when thinking about picking up a framework plan for the future in their choice. That’s why picking a scalable framework is quite essential for many, and Django is just that. It allows you to take a lot of different actions regarding scalability, such as running separate servers for the database, the media, and the application itself or even use clustering or load-balancing to distribute the application across multiple servers.5. Built-in AdminThe Django team was quite thoughtful when they created the framework, and they kept user and client satisfaction in mind. It’s quite unreasonable to create your own admin interface at the backend just to be able to manage your data with basic CRUD operations. That’s why Django offers an administrative interface right out of the box that is both professional and versatile, according to the documents the developer can now develop with the presentation in mind.

3) What is MVC and MVT pattern, what is the difference between them?

MVC (Model View Controller) and MVT (Model View Template) design patterns are and the differences between them. MVC and MVT patterns allow developers to change the visual part of an app and the business logic part separately.

MVC

It is a software design pattern that is used to implement user interfaces and gives emphasis on separating data representation from the components which interact and process the data.   
  
It has 3 components and each component has a specific purpose:

* This **Model** is the central component of this architecture and manages the data, logic as well as other constraints of the application.
* The **View** deals with how the data will be displayed to the user and provides various data representation components.
* The **Controller** manipulates the Model and renders the view by acting as a bridge between both of them.

MVT

This is yet another design pattern similar to MVC. It is also used for implementing web interfaces and applications but in contrast to MVC, the controller part is taken care for us by the framework itself.   
  
It has 3 components and each component has a specific purpose:

* This **Model** similar to MVC acts as an interface for your data and is basically the logical structure behind the entire web application which is represented by a database such as MySql, PostgreSQL.
* The **View** executes the business logic and interacts with the Model and renders the template. It accepts HTTP request and then return HTTP responses.
* The **Template** is the component which makes MVT different from MVC. Templates act as the presentation layer and are basically the HTML code that renders the data. The content in these files can be either static or dynamic.

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Model View Controller (MVC)** | **Model View Template (MVT)** |
| 1. | **MVC has controller that drives both Model and View.** | **MVT has Views for receiving HTTP request and returning HTTP response.** |
| 2. | View tells how the user data will be presented. | Templates are used in MVT for that purpose. |
| 3. | In MVC, we have to write all the control specific code. | Controller part is managed by the framework itself. |
| 4. | Highly coupled | Loosely coupled |
| 5. | Modifications are difficult | Modifications are easy |
| 6. | Suitable for development of large applications but not for small applications. | Suitable both small and large applications. |
| 7. | Flow is clearly defined thus easy to understand. | Flow is sometimes harder to understand as compared to MVC. |
| 8. | It doesn’t involve mapping of URLs. | URL pattern mapping takes place. |
| 9. | Examples are ASP.NET MVC, Spring MVC etc. | Django uses MVT pattern. |

4)What is a virtual environment ? why is it recommended to use the virtual environment in projects?

A virtual environment, often referred to as a “virtualenv” in the Python development world, is a crucial tool for Django development. It’s essentially an isolated Python environment that allows you to manage project-specific dependencies separately from the system-wide Python installation. Here’s why virtual environments are important for Django development:

1. **Isolation of Dependencies**: Virtual environments enable you to create isolated environments for different projects. This means that each Django project you work on can have its own set of libraries and dependencies without interfering with one another. This isolation ensures that changes made in one project, such as upgrading a library, won’t break another project using a different version of the same library.

1. **Version Compatibility:** Django and its related packages evolve over time, and different projects may require different versions of these packages. A virtual environment allows you to specify and manage the exact versions of Django and its dependencies for a particular project. This ensures that your project remains compatible with the versions it was developed with, even if you upgrade Django or other packages for other projects.

1. **Ease of Deployment**: When you’re ready to deploy your Django project to a production server, having a virtual environment simplifies the process. You can create a requirements.txt file that lists all the project’s dependencies along with their versions. This file can then be used to recreate the exact environment on the production server, ensuring consistency and reducing the chances of runtime errors.

1. **Cleaner Development Workflow**: With virtual environments, your development environment remains clean and uncluttered. You won’t accidentally install packages globally, which could lead to conflicts or difficulties in tracking dependencies. Virtual environments keep your system Python installation untouched, reducing the risk of breaking system-level packages.

1. **Security:** Virtual environments can enhance the security of your projects. If a project is compromised, it won’t affect other projects or your system as a whole since it operates within its isolated environment.

Virtual environments are indispensable for Django development because they provide a controlled and isolated environment for managing project-specific dependencies. They ensure version compatibility, ease deployment, promote a clean development workflow, and enhance security, making it easier to develop and maintain Django projects with confidence. To create a virtual environment, you can use Python’s built-in `venv` module or third-party tools like `virtualenv`.

Virtualenv : is a python tool used for creating isolated environments. This separates each project from another so that the tools and actions active in one project environment do not have an effect in another environment except they are installed or activated in the other environment.

1. Install Virtualenv
2. Locate where Python is installed on your PC
3. Create a virtual Python Environment for your Project
4. Activating the Virtual Environment
5. Create a Django Project