1- What is Django and why is it used?

Django (named after the Django Reinhardt) is a high-level python-based free and open-source web framework that follows the model-view-template(MVT) architectural pattern.

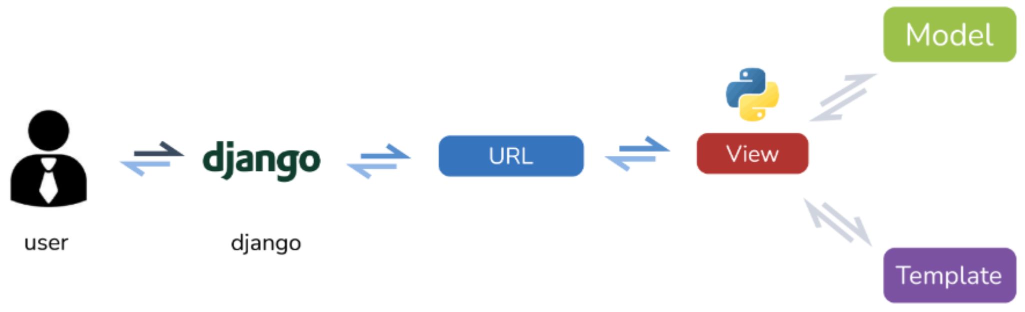
Advantages of using Django:

- Rich Ecosystem: It comes with numerous third-party apps which can be easily integrated as per the requirements of the project.
- Maturity: Django has been in use for over a decade. In the time frame, a lot of features are added and enhanced to make it a Robust framework. Apart from that, there are a large number of developers who are using Django.
- Admin panel: Django provides an admin dashboard that we can use to do basic CRUD operations over the models.
- Plugins: Allow programmers to add various features to applications and leave sufficient space for customization.
- Libraries: Due to the large development community there is an ample number of libraries for every task.
- ORM: It helps us with working with data in a more object-oriented way

2- Explain Django Architecture?

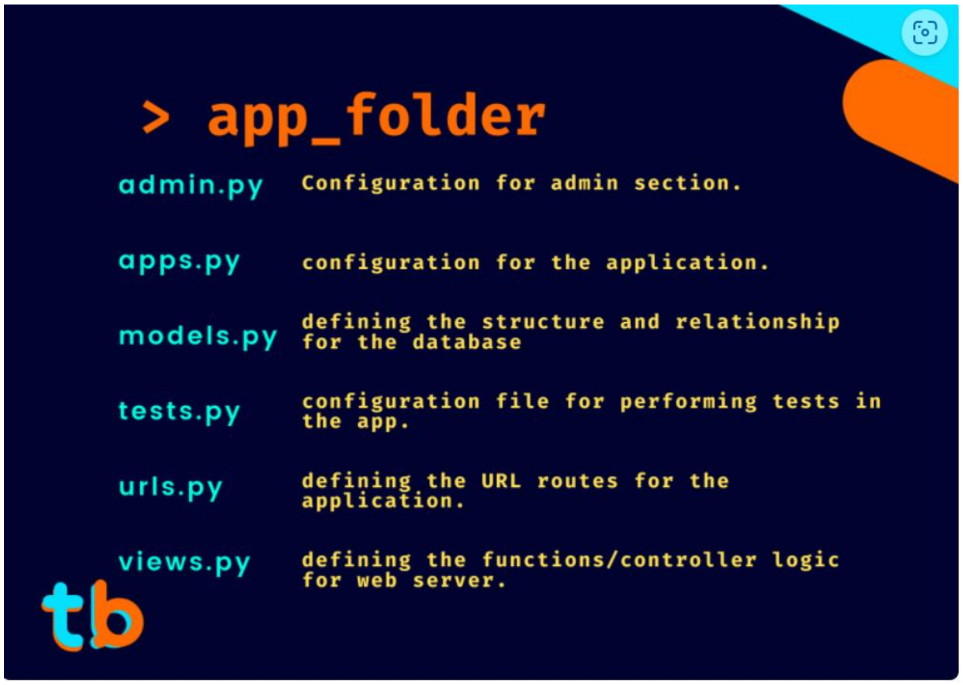
Django follows the MVT (Model View Template) pattern which is based on the Model View Controller architecture. It's slightly different from the MVC pattern as it maintains its own conventions.

- The model will serve as an interface for your data. It is in charge of data management. A database represents the logical data structure that supports the entire application such as MySql and Postgres.
- The View is the user interface, that renders a website page in your browser. HTML/CSS/Javascript and Jinja files are used to represent it.
- A template is made up of both static sections of the desired HTML output and specific syntax that describes how dynamic content will be included.



3- Explain the Django project directory structure.

- **init.py** – It's an empty Python file. It is called when the package or one of its modules is imported. This file tells the Python interpreter that this directory is a package and that the presence of the __init.py_ file makes it a Python project.
- **manage.py** – This file is used to interact with your project from the command line utility. with the help of this command, we can manage several commands such as:
  - manage.py runserver
  - manage.py makemigration
  - manage.py migrate' etc
- **setting.py –** It is the most important file in Django projects. It holds all the configuration values that your web app needs to work, i.e. pre-installed, apps, middleware, default database, API keys, and a bunch of other stuff.
- **views.py** – The View shows the user the model's data. The view knows how to get to the data in the model, but it has no idea what that data represents or what the user may do with it.
- **urls.py** – It is a universal resource locator which contains all the endpoints, we store all links of the project and functions to call it.
- **models.py** – The Model represents the models of web applications in the form of classes, it contains no logic that describes how to present the data to a user.
- **wsgi.py** – WSGI stands for Web Server Gateway Interface, This file is used for deploying the project in WSGI. It helps communication between your Django application and the web server. more…
- **admin.py –** It is used to create a superuser Registering model, login, and use the web application.
- **app.py –** It is a file that helps the user to include the application configuration for their app.



> app_folder

admin.py	Configuration for admin section.
apps.py	configuration for the application.
models.py	defining the structure and relationship for the database
tests.py	configuration file for performing tests in the app.
urls.py	defining the URL routes for the application.
views.py	defining the functions/controller logic for web server.

Other folders/files

- **Static**

  The Static folder is the folder in which you store your css, javascript, and images(images or media files that are used in the templates). This is a good way to improve the performance as in the production the webserver collects all the static files and store them in a single place for responding to the requests.

4- **How do you create a Django Project?**
D:/ProjectDirectory/ >: django-admin startproject projectname

5- **How do you create a Django App?**
D:/ProjectDirector/ProjectName: python manage.py startapp "appname"

6- Importance of virtual environment setup for Django.
A virtual environment allows you to establish separate dependencies of the different projects by creating an isolated environment that isn't related to each other and can be quickly enabled and deactivated when you're done.
It is not necessary to use a virtual environment without a virtual environment we can work with Django projects. However, using virtualenv is thought to be the ideal practice. Because it eliminates dependencies and conflicts.

7- Give a brief about the Django admin interface.
Django provides us with a default admin interface that is very helpful for creating, reading, updating, and deleting model objects that allow the user to do administrative

tasks. It reads a set of data that explains and provides information about data from the model in order to create a quick interface where the user can alter the application's contents. This is an in-built module.

8- **What are Django URLs?**

The routing of a website is determined by its URLs. In the program, we build a python module or a file called urls.py. The navigation of your website is determined by this file. When a user visits a given URL route in the browser, the URLs in the urls.py file are compared. The user then receives the response for the requested URL after retrieving a corresponding view method.

9- **What are views in Django?**

A view function, or "view" for short, is simply a Python function that takes a web request and returns a web response. This response can be HTML contents of a web page, or a redirect, or a 404 error, or an XML document, or an image, etc.
Django views are an important feature of the MVT Structure. This is a function or class that takes a web request and delivers a Web response, according to the Django script

- **Function-Based Views:** In this, we import our view as a function.
- **Class-based Views**: It's an object-oriented approach

10- **What are models in Django?**

A model in Django refers to a class that maps to a database table or database collection. Each attribute of the Django model class represents a database field. They are defined in app/models.py

```
from django.db import models

# Create your models here.

class BlogModel(models.Model):

    title = models.CharField(max_length = 1000)

    content = FroalaField()

    slug = models.SlugField(max_length = 1000,null = True,blank = True )

    user = models.ForeignKey(User,blank=True,null = True,on_delete=models.CASCADE)

    image = models.ImageField(upload_to='blog')

    created_at = models.DateTimeField(auto_now_add = True)

    upload_to = models.DateTimeField(auto_now=True)
```

11- **What do the following commands do?**

a. python manage.py makemigrations

b. python manage.py

The **makemigration command** scans the model in your application and generates a new set of migrations based on the model file modifications. This command generates the SQL statement, and when we run it, we obtain a new migration file. After running this command, no tables will be created in the database.

Running **migrate command** helps us to make these modifications to our database. The migrate command runs the SQL instructions (produced by makemigrations) and applies the database changes. After running this command, tables will be created.

12- What is Sessions?

Sessions are the technique for keeping track of a site's and a browser's state. During the session, the user data is stored in sessions, which are server-side files. The session ends when the user closes the browser or logs out of the program. Within a session, we can keep as much data as we want, We must use the session start() method to start the session. There is one advantage of using a session is that the data is stored in an encrypted format.

8. What is the Django admin site and how do you register a model with it?

The Django admin site is a built-in, powerful, and customizable administrative interface provided by the Django web framework. It's designed to make it easier for developers and administrators to manage and interact with the application's data without having to write custom administrative views and templates.

To register a model, you create an admin.py file within the app and use the admin.site.register(ModelName) method.

10.

How can you retrieve data from the database using Django's ORM?

You can retrieve data using the Django ORM by using methods like .objects.all(), .filter(), .get(), and more on a model's QuerySet. These methods generate SQL queries and return Python objects, making database interactions more intuitive.

13.

What is Django's built-in user authentication system?

Hide Answer

Django provides a comprehensive authentication system that includes features like user registration, login, logout, password reset, and user permissions. It can be easily integrated into your application to manage user authentication and access control.

14.

How can you handle forms in Django?
Django offers a form-handling framework that simplifies form creation, validation, and data processing. You can define forms using Python classes and render them in templates. The framework handles input validation and error messages.

15. *What is the significance of Django's settings.py file?*

The settings.py file contains configuration settings for a Django project. It defines database connections, middleware, installed apps, static and media file paths, and more. It centralizes project-wide settings for easy management.
19. *What is the Django Rest Framework and why is it beneficial?*

Django Rest Framework (DRF) is a powerful toolkit for building Web APIs in Django applications. It streamlines the process of designing, developing, and testing APIs by providing features like authentication, serialization, permissions, pagination, and more out of the box. DRF's class-based views and serializers simplify API development, and it supports various data formats including JSON and XML. Its authentication and permission classes ensure robust security, making it a valuable tool for creating scalable and secure APIs.

20. *How can you secure your Django application against common web vulnerabilities?*

To secure a Django application against common web vulnerabilities:

- Use parameterized queries or the ORM to prevent SQL injection.
- Implement proper input validation and sanitation to prevent XSS attacks.
- Apply authentication and authorization mechanisms to control user access.
- Use HTTPS to encrypt data in transit and prevent eavesdropping.
- Protect against CSRF attacks by using built-in CSRF tokens.
- Implement proper password hashing using Django's authentication system.
- Regularly update Django and third-party packages to patch security vulnerabilities.
- Apply content security policies (CSP) to prevent malicious scripts.
- Limit error details in production settings to avoid exposing sensitive information.

21. *Explain the purpose of Django's contrib apps.*

Django's contrib apps are reusable applications provided by the Django project. They cover common functionalities such as authentication (contrib.auth), administration (contrib.admin), sessions (contrib.sessions), and more.

22.. *How do you handle file uploads in Django?*

Django handles file uploads using the FileField and ImageField fields in models. Uploaded files are stored on the server's filesystem, with the file path stored in the database. You can use the FileUploadHandler class to customize file handling behavior, such as renaming files or defining storage locations. To display and manage uploaded files, Django's admin interface provides a user-friendly file browser.