### **1. What is Django? And why is it used?**

Django is a high-level Python web framework that enables the rapid development of secure and maintainable websites. It's free and open source. It takes care of much of the hassle of web development and allows you to focus on writing apps without any need to reinvent the wheel.

The purpose behind developing this framework is to make developers spend time on new application components instead of already developed components.

The reasons why Django is most preferred are:

* The Django framework is fast and flexible.
* Suits for any web app development
* It's secure and Scalable.
* Portable

### **2. Is Django backend or front end?**

Django is suitable for both the backend and frontend. It's a collection of Python libraries that allow you to develop useful web apps ideal for backend and frontend purposes.

### **3. What is the latest version of Django? And explain its features.**

The latest version of Django is Django 3.1. The new features of it are:

* Supports asynchronous views and middleware
* Provides JSON field support for all database backends
* Admin layout
* Path lib
* Code Reusability

CDN Integration

* SECURE\_REFERRER\_POLICY

### **4. What is the difference between Python and Django?**

Both Python and Django are intertwined but not the same. Python is a programming language used for various application developments: machine learning, artificial intelligence, desktop apps, etc.

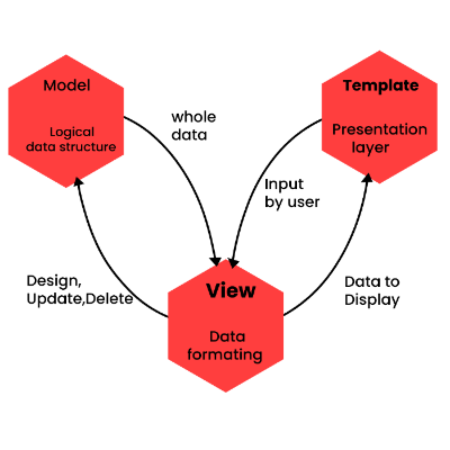
Django is a Python web framework used for full-stack app development and server development.

Using core Python, you can build an app from scratch or craft the app with Django using prewritten bits of code

### **5. What architecture does Django use?**

Django follows a Model-View-Template (MVT) architecture. It contains three different parts:

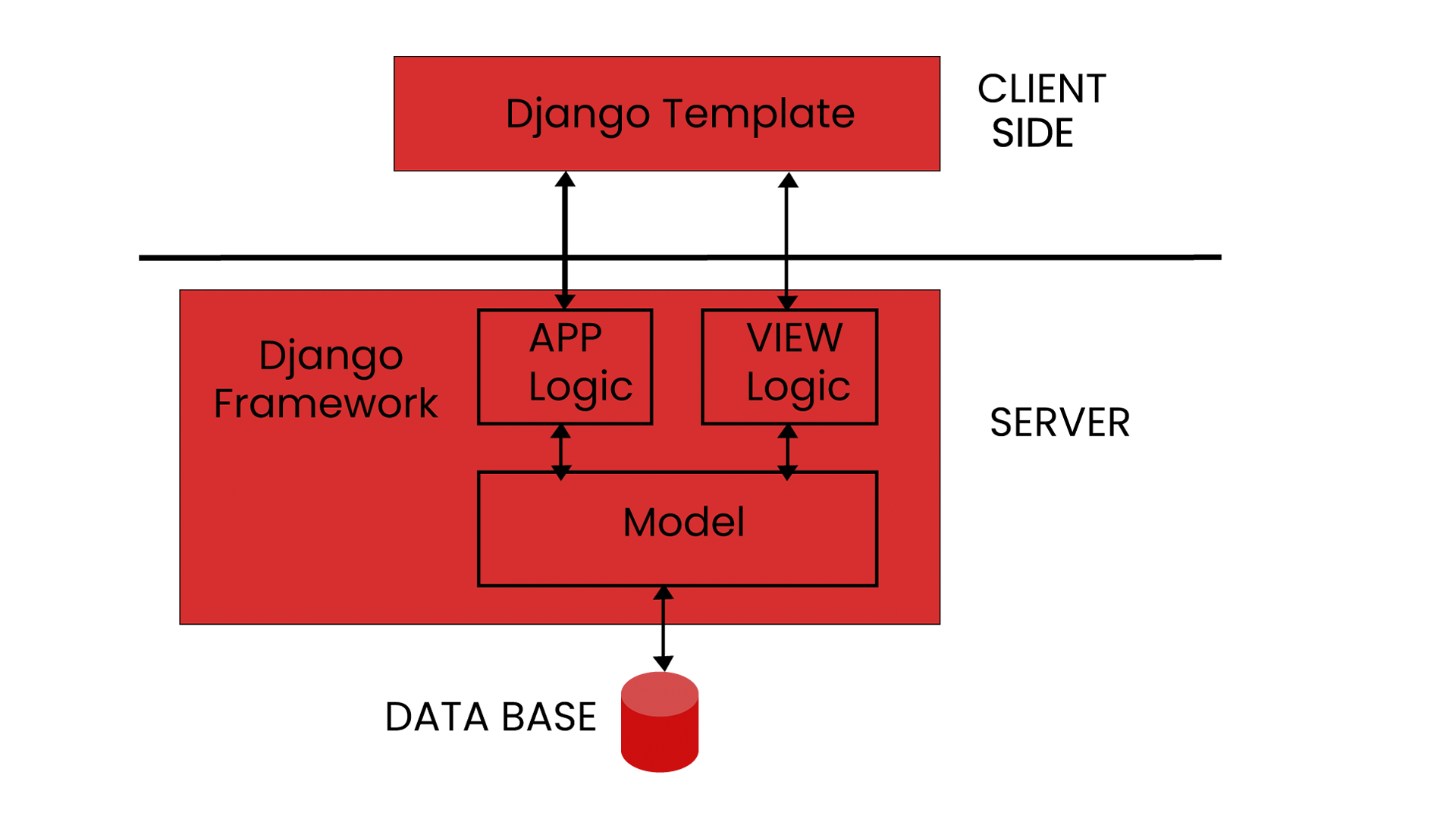
* ****Model****: Logical data structure behind the entire app and signified by a database.
* ****View:**** It's a user interface. What you see when you visit a website is called a user interface. Represented by HTML/CSS/Javascript files.
* ****Template:**** Deals with the presentation of data.



### **6. Explain Django Architecture**

As discussed in the previous question, Django follows MVT architecture - Model, Template, View.

The below diagram depicts the working cycle of Django MVT architecture:



From the diagram, you'll notice Template is on the Client side, and both the Model and View are on the Server side. Django uses request and response objects to communicate between the client and server.

If the website receives the request, it is transmitted from browser to server to manage the view file using a template.

After sending the correct URL request, the app logic and Model initiate the right response to the presented request. After that, a detailed response is sent back to View to check the response and transmit it as an HTTP response or desired user format. Then it again passes to the browser via Templates.

For your clear understanding, let's take a real-life example:

While logging into Django based website, you open the login page. It happens because View will process the request and send it to the login page URL. Then the response is sent from a server to the browser.

After then, you'll enter the credentials in Template, and the data sent back to the View to rectify the request, and then data is presented in the Model. Then the Model verifies the data provided by the user in the connected database.

If the user's data matches, it sends the related data (profile name, image, etc.) to the Views.

Otherwise, the model passes the negative result to the Views.

That's how the Django MVT architecture is working.

### **7. Explain Django's code reusability.**

Compared to other frameworks, Django offers more code reusability. As Django is a combination of apps, copying those apps from one directory to another with some tweaks to the settings.py file won't need much time to write new applications from scratch.

That is why Django is the rapid development framework, and this kind of code reusability is not allowed in any other framework.

### **8. Is Django easy to learn?**

Yes, Django is an easy-to-learn framework compared to others. Having some knowledge of Python and web-working helps you to start developing with Django.

### **9. What are the unique features of Django that make it a better framework?**

The best features of Django that make it better compared to others are:

* Compared to other open-source technologies, Django offers excellent documentation in the market.
* It's a
* [web framework](https://mindmajix.com/top-20-python-frameworks-list" \t "https://mindmajix.com/_blank) and one of the main reasons that people started using it. It's the only one that can solve any kind of operation out there.
* Django is SEO optimized.
* Django is scalable and can flexibly switch from small to large-scale projects.
* Versatile in nature. Django allows you to build applications for various types of domains.
* It has a vast community to connect with and share.
* Provides rapid development

### **10. What are the advantages of Django?**

Django has many advantages, but we'll look at major ones that differentiate it from other frameworks.

* Better CDN connectivity and content management
* Designed as batteries included framework
* Supports MVC programming paradigm
* Provides robust security features
* Accelerated custom web app development
* Compatible with major operating systems and databases

### **11. Describe the inheritance styles in Django?**

Django offers three inheritance styles:

1. ****Abstract base classes:**** You use this style when you want the parent class to retain the data you don't want to type out for every child model.
2. ****Multi-table inheritance:****You use this style when you want to use a subclass on an existing model and want each model to have its database table.
3. ****Proxy models:**** You use this style to modify Python-level behaviour with the models without changing the Model's field.

### **12. What are Django Models?**

A model is a definitive source of information about data, defined in the “app/models.py”.

Models work as an abstraction layer that structures and manipulates data. Django models are a subclass of the "django.db.models". Model class and the attributes in the models represent database fields.

### **13. Give a brief about the settings.py file.**

As the name implies, it's the main settings file of the Django file. Everything inside the Django project, like databases, middlewares, backend engines, templating engines, installed applications, static file addresses, main URL configurations, allowed hosts and servers, and security key stores in this file as a dictionary or list.

So when Django files start, it first executes the settings.py file and then loads the respective databases and engines to quickly serve the request.

### **14. Is Django a CMS?**

No, Django is not CMS (Content Management System). It's just a web framework and programming tool that allows you to build websites.

### **15. What are static files in Django? And how can you set them?**

In Django, static files are the files that serve the purpose of additional purposes such as images, CSS, or JavaScript files. Static files managed by “django.contrib.staticfiles”. There are three main things to do to set up static files in Django:

1) Set STATIC\_ROOT in settings.py

2) Run manage.py collect static

3) Set up a Static Files entry on the PythonAnywhere web tab

### **16. What is the use of Middlewares in Django?**

Middlewares in Django is a lightweight plugin that processes during request and response execution. It performs functions like security, CSRF protection, session, authentication, etc. Django supports various built-in middlewares.

### **17. What is the difference between CharField and TextField in Django?**

* TextField is a large text field for large-sized text. In Django, TextField is used to store paragraphs and all other text data. The default form widget for this field is TextArea.
* CharField is a string field used for small- to large-sized strings. It is like a string field in C/C++. In Django, CharField is used to store small strings like first name, last name, etc.

### **18. Describe Django Field Class types?**

Every field in a model is an instance of the appropriate field class. In Django, field class types determine:

* The column type describes the database about what kind of data to store (e.g., INTEGER, VARCHAR, TEXT).
* The default HTML widget, while rendering a form field

(e.g. <input type="text">, <select>)

* The minimal validation requirements used in automatically generated forms and Django admin.

### **19. What is the usage of "Django-admin.py" and "manage.py"?**

* Django-admin.py - It is a command-line utility for administrative tasks.
* manage.py - It is automatically created in each Django project and controls the Django project on the server or even to begin one. It has the following usage:

1. Manages the project's package on the sys. path.
2. Sets the DJANGO\_SETTINGS\_MODULE environment variable

### **20. What are signals in Django?**

Django includes a "signal dispatcher" to notify decoupled applications when some action takes place in the framework. In a nutshell, signals allow specific senders to inform a suite of receivers that some action has occurred. They are instrumental when we use more pieces of code in the same events.

Django provides a set of built-in signals that enable users to get notified of specific actions.

|  |  |
| --- | --- |
| ****Signal**** | ****Description**** |
| Django.db.models.signals.pre\_save(or)django.db.models.signals.post\_save | Sent before or after a model’s save() method calls. |
| django.db.models.signals.pre\_delete  (or)django.db.models.signals.post\_delete | Sent before or after a model’s delete() method or query set’s delete() method calls. |
| django.db.models.signals.m2m\_changed | We use this signal when ManyToManyField on a model changes. |
| Django.core.signals.request\_started(or)django.core.signals.request\_finished | We use this signal when Django starts or finishes an HTTP request. |

## **Django Interview Questions for Experienced**

### **21. What’s the difference between a project and an app in Django?**

The app is a module that deals with the dedicated requirements in a project. On the other hand, the project covers an entire app. In Django terms, a project can contain different apps, while an app features in various projects.

### **22. Explain Django URL in brief?**

Django allows you to design URL functions however you want. For this, you need to create a Python module informally called URLconf (URL configuration).

This module is purely a Python code and acts as a mapping between URL path expressions and [Python functions](https://mindmajix.com/python-string-functions" \t "https://mindmajix.com/_blank). Also, this mapping can be as long or short as needed and can also reference other mappings.

The length of this mapping can be as long or short as required and can also reference other mappings. Django also provides a way to translate URLs according to the active language.

### **23. What are Django Exceptions?**

An exception is an abnormal event that leads to program failure. Django uses its exception classes and python exceptions as well to deal with such situations.

We define Django core exceptions in "Django.core.exceptions". The following classes are present in this module:

|  |  |
| --- | --- |
| ****Exception**** | ****Description**** |
| AppRegistryNotReady | This class raises for using models before loading the app process. |
| ObjectDoesNotExist | It’s a base class for DoesNotExist exceptions. |
| EmptyResultSet | This exception arises when the query fails to return results. |
| FieldDoesNotExist | When the requested file does not exist, this exception arises. |
| MultipleObjectsReturned | It raises by the query multiple objects returned when we expect only one object. |
| SuspiciousOperation | It raises when the user has performed some operation, which is considered suspicious from a security perspective. |
| PermissionDenied | It arises when a user does not have permission to execute a specific action requested. |
| ViewDoesNotExist | When the requested view does not exist, this exception raises. |
| MiddlewareNotUsed | When there is no middleware in server configuration, this exception arises. |
| ImproperlyConfigured | When Django configuration is improper, this exception arises. |
| FieldError | When there is a problem with the model field, this exception arises. |
| ValidationError | It raises when data validation fails. |

### **24. Explain Django session**

Django uses the session to keep track of the state between the site and a particular browser. Django supports anonymous sessions. The session framework stores and retrieves data on a per-site-visitor basis. It stores the information on the server side and supports sending and receiving cookies. Cookies store the data of session ID but not the actual data itself.

### **25. What are Django cookies?**

A cookie is a piece of information stored in the client's browser. To set and fetch cookies, Django provides built-in methods. We use the set\_cookie() method for setting a cookie and the get() method for getting the cookie.

You can also use the request.COOKIES['key'] array to get cookie values.

### **26. Flask vs. Django: What's the difference between Flask & Django?**

Flask and Django are the two most popular Python web frameworks. The following table lists some significant differences between Django and Flask

|  |  |  |
| --- | --- | --- |
| ****Comparison Factor**** | ****Django**** | ****Flask**** |
| created | Django is a web development framework for Python. Its created in 2005 | Flask is a web microframework offering basic features of web apps. Its created in 2010 |
| Project Type | High-level Python web framework for easy and simple projects. | Low-level Python web framework. |
| Features | The best features of Django are open-source, rapid development, robust documentation, great community, and easy to learn. | The best features of Flask are open source, lightweight, and require less code to develop an app. |
| Type of Framework | Full-stack web framework | WSGI (Web Server Gateway Interface ) framework |
| Templates, Admin, and ORM | Built-in | Requires installation |
| Flexibility | Django Web Framework supports a large number of third-party applications. | Flask Web Framework doesn't offer support for third-party applications. |
| Companies using | Instagram, Coursera, Udemy. | Netflix, Reddit, Lyft, MIT |
| Visual Debugging | Django does not support visual debugging. | Flask supports visual debugging. |
| Bootstrapping tool | Builtin | Not available |
| Working style | Offers monolithic working style | Offers diversified working style |
| Project layout | The structure of the project layout is conventional. | The structure of the project layout for the flask is random. |

### **27. How to check the version of Django installed on your system?**

To check the version of [Django installed on your system](https://docs.djangoproject.com/en/3.1/topics/install/" \t "https://mindmajix.com/_blank), open the command prompt and enter the following command:

py -m django --version

You can also try to import Django and use the get\_version() method as follows:

import django

print(django.get\_version())

### **28. Give a brief about Django Admin.**

Django Admin is the command-line utility for administrative tasks. It's a preloaded interface to fulfill all web developer's needs and is imported from the "django.contrib packages".

Django Admin interface has its user authentication and offers advanced features like authorizing the access, CMS (Content Management System), managing various models, etc.

You can even perform the following tasks using Django admin as listed out in the table:

|  |  |
| --- | --- |
| ****Command**** | ****Task**** |
| django-admin help | Displays the usage of the information and commands list provided by each application. |
| django-admin help –command | Displays available commands |
| django-admin help <command> | Displays the command description and its available options |
| django-admin version | Determines Django’s version |
| django-admin make migrations | Depending on the changes done in the model creates new migrations |
| django-admin migrate | Synchronizes the database state with the present set of models and migrations |
| django-admin runserver | Starts the development server |
| django-admin sendtestemail | A test mail sent to confirm Django email working status |
| django-admin shell | Starts the Python interactive interpreter |
| django-admin showmigrations | Displays all the project’s migrations |

### **29. How do you create a Django project?**

To create a Django project, navigate to the directory where you want to do a project and type the following command:

$ django-admin startproject ABC

That will create an "ABC" folder with the following structure −

ABC/

manage.py

myproject/

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

Note: Here, "ABC" is the name of the project. You can mention any name you want.

### **30. Name some companies using Django.**

Various companies out there are using Django. Of them, major are Instagram, Pinterest, Udemy, Mozilla Firefox, Reddit, etc.

### **31. How do Django views work?**

Django views are the critical component of the framework They serve the purpose of encapsulation. They encapsulate the logic liable to process a user's request and return a response to the user.

Either they return HTTP responses or raise an exception such as 404 in Django. Besides, Views also perform tasks like reading records from a database, generating PDF files, etc.

Every app in Django comes with a views.py file, and this contains the views functions. Views function can be imported directly in the URLs file in Django.

To achieve that, you have to import the view function in the urls.py file first and add the path/URL that the browser should request to call that View function.

|  |
| --- |
| Explore **[Python Interview Questions](https://mindmajix.com/python-interview-questions" \o "Python Interview Questions" \t "https://mindmajix.com/_blank)** that help you grab high-paying jobs. |

### **32. Give a brief about Django Template?**

Django Templates generate dynamic web pages. Using templates, you can show the static data and the data from various databases connected to the app through a context dictionary. You can create any number of templates based on project requirements. Even it's OK to have none of them.

Django template engine handles the templating in the Django web framework. Some template syntaxes declare variables, filters, control logic, and comments.

Django ships built-in backends for its template system called the Django template language (DTL).

### **33. Describe Django ORM.**

In Django, the most notable feature is Object-Relational Mapper (ORM), which allows you to interact with app data from various relational databases such as SQLite, MySQL, and PostgreSQL.

Django ORM is the abstraction between web application data structure (models) and the database where the data is stored. Without writing any code, you can retrieve, delete, save, and perform other operations over the database.

The main advantage of ORMs is rapid development. ORMs make projects more portable. It's easier to change the database with Django ORM.

### **34. When to use iterators in Django ORM?**

Iterators are containers in Python containing several elements. Every object in the iterator implements two methods that are \_\_init\_\_() and the \_\_next\_\_() methods.

In Django, the fair use of an iterator is when you process results that take up a large amount of memory space. For this, you can use the iterator() method, which evaluates the QuerySet and returns the corresponding iterator over the results.

### **35. What is Django caching? And explain the strategies used to implement it.**

Caching is the process of saving expensive calculation output to avoid performing the same calculation again.

Django supports a robust cache system to save web pages such that they don't have to be evaluated repeatedly for each request.

They are few strategies to implement caching in Django, and the following table lists them:

|  |  |
| --- | --- |
| ****Strategy**** | ****Description**** |
| Memcached | The most efficient and faster memory-based cache server |
| Filesystem caching | Cache files store in serial order in separate files. |
| Local-memory caching | If you have not specified any other, this is the default cache. It’s per-process and threads safe as well. |
| Database caching | Cache data will be stored in the database and works OK if you have a well-indexed database server. |

### **36. How does Django process a request?**

Whenever the Django Server receives a request, the system follows an algorithm to determine which Python code needs execution. Here are the steps that sum up the algorithm:

* Django checks the root URL configuration.
* Next, Django looks at all the variable URL patterns in the URLconf for the match of the requested URL
* If the URL matches, it returns the associated view function.
* It will then request the data from the Model of that app for any data requirement and pass it to the corresponding Template rendered by the browser.
* Django sends an error-handling view if none of the URLs match the requested URL.

### **37. Which Python version should be used with Django?**

Python 3 is the most recommended version for Django. Because it's faster, has more features, and is better supported.

### **38. Explain the file structure of a typical Django project.**

A typical Django project consists of these four files:

* manage.py
* settings.py
* \_\_init\_\_.py
* urls.py
* wsgi.py

The final four files are inside a directory, which is at the same level as manage.py.

* manage.py is the command-line utility of your Django project and controls the Django project on the server.
* settings.py file includes information on all the apps installed in the project.
* The urls.py file acts as a map for the whole web project.
* The \_\_init\_\_.py file is an empty file that makes the python interpreter understand that the directory consisting of settings.py is a module/ package.
* The wsgi.py file is for the server format WSGI

### **39. Why is Django called a loosely coupled framework?**

Django is known as a loosely coupled framework beca+use of its MVT architecture.

Django's architecture is a variant of MVC architecture, and MVT is beneficial because it completely discards server code from the client's machine. Models and Views are present on the client machine, and templates only return to the client.

All the architecture components are different from each other. Both frontend and backend developers can work simultaneously on the projects as they won't affect each other when changed.

### **40. What is the Django REST framework (DRF)?**

Django REST framework is a flexible and powerful toolkit for building Web APIs rapidly.

The following are the significant reasons that are making REST framework perfect choice:

* Web browsable API
* Authentication policies
* [Serialization](https://mindmajix.com/python/serialization" \t "https://mindmajix.com/_blank)
* Extensive documentation and excellent community support.
* Perfect for web apps since they have low bandwidth.
* Global companies like Red Hat, Mozilla, Heroku, Eventbrite, etc., trust this framework.

## **Advanced Django Interview Questions**

### **41. Is Django too monolithic? Explain this statement.**

The Django framework is monolithic, which is valid to some extent. As Django's architecture is MVT-based, it requires some rules that developers need to follow to execute the appropriate files at the right time.

With Django, you get significant customizations with implementations. Through this, you cannot change file names, variable names, and predefined lists.

Django's file structure is a logical workflow. Thus the monolithic behavior of Django helps developers to understand the project efficiently.

### **42. Explain user authentication in Django**

Django comes with a built-in user authentication system to handle objects such as users, groups, permissions, etc. It not only performs authentication but authorization as well.

Following are the system objects:

* users
* Groups
* Password Hashing System
* Permissions
* A pluggable backend system
* Forms Validation

Apart from this, there are various third-party web apps that we can use instead of the default system to provide more user authentication with more features.

### **43. What is the "django.shortcuts.render" function?**

When a View function returns a web page as HttpResponse instead of a simple string, we use the render function.

Render is a shortcut for passing a data dictionary with a template. This function uses a templating engine to combine templates with a data dictionary.

Finally, the render() returns the HttpResponse with the rendered text, the models' data.

Syntax:

render(request, template\_name, context=None, content\_type=None, status=None, using=None)

The request generates a response.

The template name and other parameters pass the dictionary.

For more control, specify the content type, the data status you passed, and the render you are returning.

### **44. What is the use of forms in Django?**

Forms serve the purpose of receiving user inputs and using that data for logical operations on databases. Django supports form class to create HTML forms. It defines a form and how it works and appears.

Django's forms handle the following parts:

* Prepares and restructures data to make it ready for rendering
* Creates HTML forms for the data
* Processes submitted forms and data from the client.

### **45. Can you explain how to add View functions to the urls.py file?**

There are two ways to add the view function to the main URLs config:

1. Adding a function View

In this method, import the particular View's function and add the specific URL to the URL patterns list.

2. Adding a Class-based view

This one is a more class-based approach. For this, import the class from the views.py and then add the URL to the URL patterns. An inbuilt method is needed to call the class as a view.

Write the name of the function on the previous method as shown below:

class\_name.as\_view()

This will pass your view class as a view function.

Both function-based and class-based have their advantages and disadvantages. Depending on the situation, you can use them to get the right results.

### **46. Explain Django Security.**

Protecting user's data is an essential part of any website design. Django implements various sufficient protections against several common threats. The following are Django's security features:

* Cross-site scripting (XSS) protection
* SQL injection protection
* Cross-site request forgery (CSRF) protection
* Enforcing SSL/HTTPS
* Session security
* Clickjacking protection
* Host header validation

### **47. What is Ajax in Django?**

AJAX (Asynchronous JavaScript And XML) allows web pages to update asynchronously to and from the server by exchanging data in Django. That means without reloading a complete webpage you can update parts of the web page.

It involves a combination of a browser built-in XMLHttpRequest object, HTML DOM, and JavaScript.

### **48. How to handle Ajax requests in Django?**

To handle Ajax requests in the Django web framework, perform the following:

* Initialize Project
* Create models
* Create views
* Write URLs
* Carry out requests with Jquery Ajax.
* Register models to admin

### **49. What are Django generic views?**

Writing views is a heavy task. Django offers an easy way to set Views called Generic Views. They are classes but not functions and stored in "django.views.generic".

Generic views act as a shortcut for common usage patterns. They take some common idioms and patterns in view development and abstract them to write common views of data without repeating yourself quickly.

### **50. What is the correct way to make a variable available to all your templates?**

In case all your templates need the same objects, use "RequestContext." This method takes HttpRequest as the first parameter and populates the context with a few variables simultaneously as per the engine's context\_processors configuration option.

## **Django FAQ**

### **51. Tell me how to use a file-based session?**

For this, we have to set the SESSION\_ENGINE settings to “Django.contrib.sessions.backends.file.”

### **52. What command-line loads data in Django?**

“Django-admin.py load data” loads data in Django. This command line performs data searching and loads the contents of the named fixtures into the database.

### **53. What is CRUD?**

CRUS is an acronym for Create, Read, Update, and Delete. It’s a mnemonic framework used for constructing models when building application programming interfaces (APIs).

### **54. Explain Django’s Request/Response Cycle.**

When a process starts, the Django server receives a request and checks for a matching URL in the project-defined URL patterns. If the URL matches, it executes the associated code in the view file with the URL and sends a response. If the server can’t find a matching URL, it invokes a 404-status code.

### **55. What do you use middleware for in Django?**

For the following functions, you can use Middleware in Django:

* Cross-site request forgery protection
* Content Gzipping
* User authentication
* Session management

### **56. Does Django support multiple-column primary keys?**

Django does not support multiple-column primary keys. It only supports single-column primary keys.

### **57. What is a QuerySet?**

In the context of Django, QuerySet is a set of SQL queries. To see the SQL query from the Django filter call, type the command print(b.query).

### **58. How to check the raw SQL queries running in Django??**

Make sure that the DEBUG setting is set to True, and type the following commands:

* from Django.db import connection
* connection.queries

### **59. Are Django signals asynchronous?**

No, Django signals are synchronous. There is no background thread or asynchronous jobs to execute them. When we use signals in applications, they allow you to maintain the code to understand application behavior and solve issues faster and better.

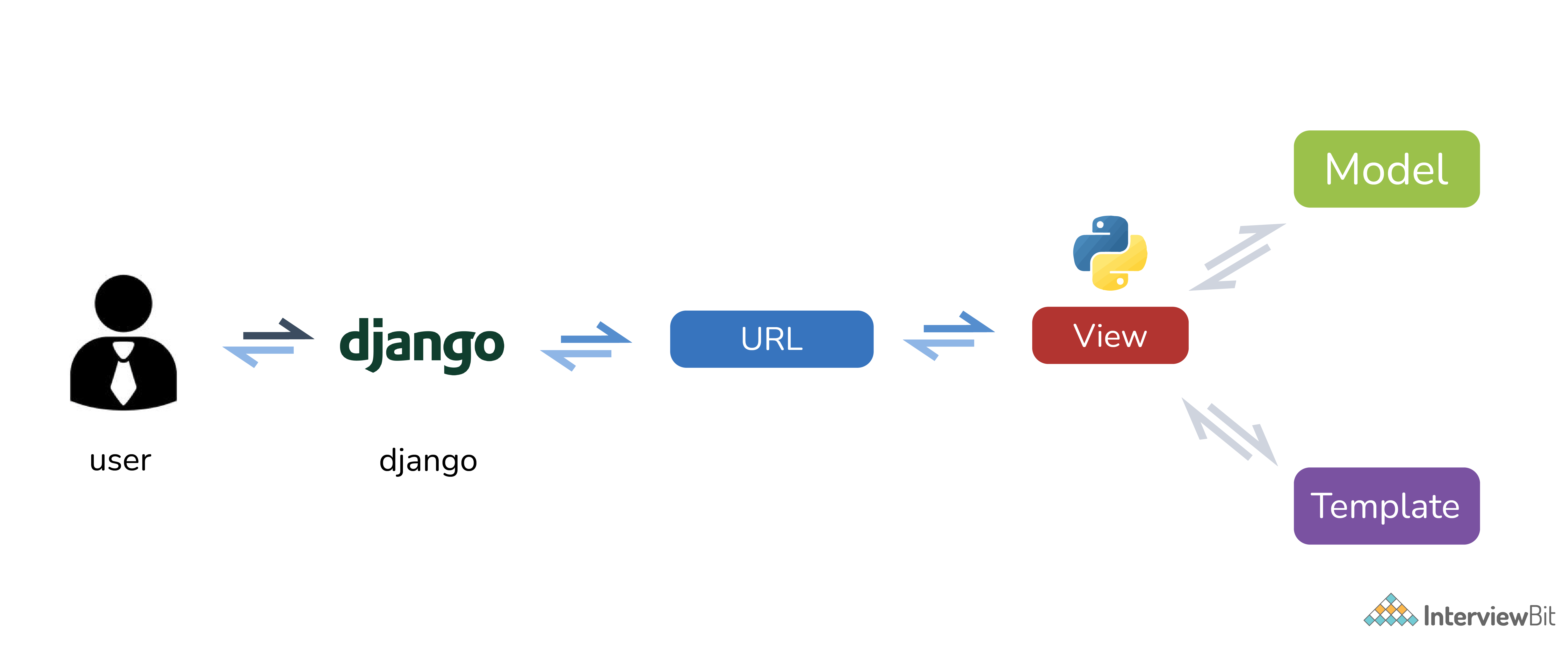
### **60. What are the Django disadvantages?**

Not suitable for small projects due to its monolithic size

* Everything connects on Django’s ORM.
* Everything must be defined explicitly due to a lack of convention.
* Django web framework has a steep learning curve.

### **1. 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, so, the controller is handled by the framework itself. The template is a presentation layer. It is an HTML file mixed with Django Template Language (DTL). The developer provides the model, the view, and the template then maps it to a URL, and finally, Django serves it to the user.



### **2. Explain the django project directory structure?**

* manage.py - A command-line utility that allows you to interact with your Django project
* \_\_init\_\_.py - An empty file that tells Python that the current directory should be considered as a Python package
* settings.py - Comprises the configurations of the current project like DB connections.
* urls.py - All the URLs of the project are present here
* wsgi.py - This is an entry point for your application which is used by the web servers to serve the project you have created.

### **3. 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

Example:

**from** django.db **import** models**class** **SampleModel**(models.Model):

field1 = models.CharField(max\_length = 50)

field2 = models.IntegerField()**class** **Meta**:

db\_table = “sample\_model”

Every model inherits from django.db.models.Model  
  
Our example has 2 attributes (1 char and 1 integer field), those will be in the table fields.

The metaclass helps you set things like available permissions, singular and plural versions of the name, associated database table name, whether the model is abstract or not, etc.  
  
To get more information about models you can refer here: [https://docs.djangoproject.com/en/3.1/topics/db/models/](https://docs.djangoproject.com/en/3.1/topics/db/models/" \t "https://www.interviewbit.com/django-interview-questions/_blank).

**You can download a PDF version of Django Interview Questions.**

**[IMG_257Download PDF](javascript:void(0))**

### **4. What are templates in Django or Django template language?**

Templates are an integral part of the Django MVT architecture. They generally comprise HTML, CSS, and js in which dynamic variables and information are embedded with the help of views. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags.

A template is rendered with a context. Rendering just replaces variables with their values, present in the context, and processes tags. Everything else remains as it is.

The syntax of the Django template language includes the following four constructs :

* Variables
* Tags
* Filters
* Comments

To read more about templates you can refer to this: [https://docs.djangoproject.com/en/3.1/topics/templates/](https://docs.djangoproject.com/en/3.1/topics/templates/" \t "https://www.interviewbit.com/django-interview-questions/_blank)

### **5. 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.

Example:

**from** django.http **import** HttpResponse**def** **sample\_function**(request):

**return** HttpResponse(“Welcome to Django”)

There are two types of views:

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

### **6. What is Django ORM?**

This ORM (an acronym for Object Relational Mapper) enables us to interact with databases in a more pythonic way like we can avoid writing raw queries, it is possible to retrieve, save, delete and perform other operations over the database without ever writing any SQL query. It works as an abstraction layer between the models and the database.

### **7. Define static files and explain their uses?**

Websites generally need to serve additional files such as images. Javascript or CSS. In Django, these files are referred to as “static files”, Apart from that Django provides django.contrib.staticfiles to manage these static files.

### **8. What is Django Rest Framework(DRF)?**

Django Rest Framework is an open-source framework based upon Django which lets you create RESTful APIs rapidly.

### **9. What is django-admin and manage.py and explain its commands?**

django-admin is Django’s command-line utility for administrative tasks. In addition to this, a manage.py file is also automatically created in each Django project. Not only does it perform the same purpose as the django-admin but it also sets the DJANGO\_SETTINGS\_MODULE environment variable to point to the project's settings.py file.

* django-admin help - used to display usage information and a list of the commands provided by each application.
* django-admin version - used to check your Django version.
* django-admin check - used to inspect the entire Django project for common problems.
* django-admin compilemessages - Compiles .po files created by makemessages to .mo files for use with the help of built-in gettext support.
* django-admin createcachetable - Creates the cache tables for use in the database cache backend.
* django-admin dbshell - Runs the command-line client for the database engine specified in your ENGINE setting(s), with the connection parameters (USER, PASSWORD, DB\_NAME, USER etc.) specified settings file.
* django-admin diffsettings - Shows the difference between the existing settings file and Django’s default settings.
* django-admin dumpdata - Used to the dumpdata from the database.
* django-admin flush - Flush all values from the database and also re-executes any post-synchronization handlers specified in the code.
* django-admin inspectdb - It generates django models from the existing database tables.
* django-admin loaddata - loads the data into the database from the fixture file.
* django-admin makemessages - Used for translation purpose and it generates a message file too.
* django-admin makemigrations - Generates new migrations as per the changes detected to your models.
* django-admin migrate - Executes SQL commands after which the database state with the current set of models and migrations are synchronized.
* django-admin runserver - Starts a light-weight Web server on the local machine for development. The default server runs on port 8000 on the IP address 127.0.0.1. You can pass a custom IP address and port number explicitly if you want.
* django-admin sendtestemail - This is used to confirm email sending through Django is working by sending a test email to the recipient(s) specified.
* django-admin shell - Starts the Python interactive interpreter.
* django-admin showmigrations - Shows all migrations present in the project.
* django-admin sqlflush - Prints the SQL statements that would be executed for the flush command mentioned above.
* django-admin sqlmigrate - Prints the SQL statement for the named migration.
* django-admin sqlsequencereset - output the SQL queries for resetting sequences for the given app name(s).
* django-admin squashmigrations - Squashes a range of migrations for a particular app\_label.
* django-admin startapp - Creates a new Django app for the given app name within the current directory or at the given destination.
* django-admin startproject - Creates a new Django project directory structure for the given project name within the current directory or at the given destination.
* django-admin test - Runs tests for all installed apps.
* django-admin testserver - Runs a Django development server (which is also executed via the runserver command) using data from the given fixture(s).
* django-admin changepassword - offers a method to change the user's password.
* django-admin createsuperuser - Creates a user account with all permissions(also known as superuser account).
* django-admin remove\_stale\_contenttypes - removes stale content types (from deleted models) in your database.
* django-admin clearsessions - Can be used to clean out expired sessions or as a cron job.

### **10. What is Jinja templating?**

Jinja Templating is a very popular templating engine for Python, the latest version is Jinja2.

Some of its features are:

* Sandbox Execution - This is a sandbox (or a protected) framework for automating the testing process
* HTML Escaping - It provides automatic HTML Escaping as <, >, & characters have special values in templates and if using a regular text, these symbols can lead to XSS Attacks which Jinja deals with automatically.
* Template Inheritance
* Generates HTML templates much faster than default engine
* Easier to debug as compared to the default engine.

### **11. What are Django URLs?**

URLs are one of the most important parts of a web application and Django provides you with an elegant way to design your own custom URLs with help of its module known as URLconf (URL Configuration). The basic functionality of this python module is to   
You can design your own URLs in Django in the way you like and then map them to the python function (View function). These URLs can be static as well as dynamic. These URLs as present in the urls.py where they are matched with the equivalent view function.

Basic Syntax:

**from** django.urls **import** path**from** . **import** views

urlpatterns = [

path('data/2020/', views.data\_2020),

path('data/<int:year>/', views.data\_year)

]

### **12. What is the difference between a project and an app in Django?**

In simple words Project is the entire Django application and an app is a module inside the project that deals with one specific use case.   
For eg, payment system(app) in the eCommerce app(Project).

### **13. What are different model inheritance styles in the Django?**

* ****Abstract Base Class Inheritance****: Used when you only need the parent class to hold information that you don’t want to write for each child model.
* ****Multi-Table Model Inheritance:****  Used when you are subclassing an existing model and need each model to have its own table in the database.
* ****Proxy Model Inheritance:****  Used when you want to retain the model's field while altering the python level functioning of the model.

## **Intermediate Django Interview Questions**

### **14. What are Django Signals?**

Whenever there is a modification in a model, we may need to trigger some actions.   
Django provides an elegant way to handle these in the form of signals. The signals are the utilities that allow us to associate events with actions. We can implement these by developing a function that will run when a signal calls it.

****List of built-in signals in the models:****

| **Signals** | **Description** |
| --- | --- |
| django.db.models.pre\_init & django.db.models.post\_init | Sent before or after a models’s \_init\_() method is called |
| django.db.models.signals.pre\_save & django.db.models.signals.post\_save | Sent before or after a model’s save() method is called |
| django.db.models.signals.pre\_delete & django.db.models.signals.post\_delete | Sent before or after a models’ delete() method or queryset delete() method is called |
| django.db.models.signals.m2m\_changed | Sent when a ManyToManyField is changed |
| django.core.signals.request\_started & django.core.signals.request\_finished | Sent when an HTTP request is started or finished |

### **15. Explain the caching strategies in the Django?**

Caching refers to the technique of storing the output results when they are processed initially so that next time when the same results are fetched again, instead of processing again those already stored results can be used, which leads to faster accessing as well us less resource utilization. Django provides us with a robust cache system that is able to store dynamic web pages so that these pages don’t need to be evaluated again for each request.   
  
Some of the caching strategies in Django are listed below:

| **Strategy** | **Description** |
| --- | --- |
| Memcached | A memory-based cache server is the fastest and most efficient |
| FileSystem Caching | Values of the cache are stored as separate files in a serialized order |
| Local-memory Caching | This is used as the default cache strategy by Django if you haven’t set anything. It is per-process as well as thread-safe. |
| Database Caching | Cache data will be stored in the database and works very well if you have a fast and well-indexed DB server. |

### **16. Explain user authentication in Django?**

Django comes with a built-in user authentication system, which handles objects like users, groups, user-permissions, and few cookie-based user sessions. Django User authentication not only authenticates the user but also authorizes him.  
  
The system consists and operates on these objects:

* Users
* Permissions
* Groups
* Password Hashing System
* Forms Validation
* A pluggable backend system

### **17. How to configure static files?**

Ensure that django.contrib.staticfiles is added to your INSTALLED\_APPS

In your settings file. define STATIC\_URL for ex.

STATIC\_URL = '/static/'

In your Django templates, use the static template tag to create the URL for the given relative path using the configured STATICFILES\_STORAGE.

{% load static %}

<img src="{% static 'my\_sample/abcxy.jpg' %}" alt="ABC image">

Store your static files in a folder called static in your app. For example my\_sample/static/my\_sample/abcxy.jpg

### **18. Explain Django Response lifecycle?**

Whenever a request is made to a web page, Django creates an HttpRequest object that contains metadata about the request. After that Django loads the particular view, passing the HttpRequest as the first argument to the view function. Each view will be returning an HttpResponse object.  
On the big picture following steps occur when a request is received by Django:

1. First of the Django settings.py file is loaded which also contain various middleware classes (MIDDLEWARES)
2. The middlewares are also executed in the order in which they are mentioned in the MIDDLEWAREST
3. From here on the request is now moved to the URL Router, who simply gets the URL path from the request and tries to map with our given URL paths in the urls.py.
4. As soon as it has mapped, it will call the equivalent view function, from where an equivalent response is generated
5. The response also passes through the response middlewares and send back to the client/browser.

### **19. What databases are supported by Django?**

PostgreSQL and MySQL, SQLite and Oracle. Apart from these, Django also supports databases such as ODBC, Microsoft SQL Server, IBM DB2, SAP SQL Anywhere, and Firebird using third-party packages. Note: Officially Django doesn’t support any no-SQL databases.

### **20. What's the use of a session framework?**

Using the session framework, you can easily store and retrieve arbitrary data based on the pre-site-visitors. It stores data on the server-side and takes care of the process of sending and receiving cookies. These cookies just consist of a session ID, not the actual data itself unless you explicitly use a cookie-based backend.

### **21. What’s the use of Middleware in Django?**

Middleware is something that executes between the request and response. In simple words, you can say it acts as a bridge between the request and response. Similarly In Django when a request is made it moves through middlewares to views and data is passed through middleware as a response.

### **22. What is context in the Django?**

Context is a dictionary mapping template variable name given to Python objects in Django. This is the general name, but you can give any other name of your choice if you want.

### **23. What is django.shortcuts.render function?**

When a view function returns a webpage as HttpResponse instead of a simple string, we use render(). Render function is a shortcut function that lets the developer easily pass the data dictionary with the template. This function then combines the template with a data dictionary via templating engine. Finally, this render() returns as HttpResponse with the rendered text, which is the data returned by models. Thus, Django render() bypasses most of the developer’s work and lets him use different template engines.  
The basic syntax:  
render(request, template\_name, context=None, content\_type=None, status=None, using=None)  
The request is the parameter that generates the response. The template name is the HTML template used, whereas the context is a dict of the data passed on the page from the python. You can also specify the content type, the status of the data you passed, and the render you are returning.

### **24. What’s the significance of the settings.py file?**

As the name suggests this file stores the configurations or settings of our Django project, like database configuration, backend engines, middlewares, installed applications, main URL configurations, static file addresses, templating engines, main URL configurations, security keys, allowed hosts, and much more.

### **25. How to view all items in the Model?**

ModelName.objects.all()

### **26. How to filter items in the Model?**

ModelName.objects.filter(field\_name=”term”)

## **Advanced Django Interview Questions**

### **27. How to use file-based sessions?**

To use the same, you need to set the SESSION\_ENGINE settings to "django.contrib.sessions.backends.file"

### **28. What is mixin?**

Mixin is a type of multiple inheritances wherein you can combine behaviors and attributes of more than one parent class. It provides us with an excellent way to reuse code from multiple classes. One drawback of using these mixins is that it becomes difficult to analyze what a class is doing and which methods to override in case of its code being too scattered between multiple classes.

### **29. What is Django Field Class?**

'Field' refers to an abstract class that represents a column in the database table.   
The Field class is just a subclass of RegisterLookupMixin. In Django, these fields are used to create database tables (db\_types()) which are used to map Python types to the database using get\_prep\_value() and the other way round using from\_db\_value() method. Therefore, fields are fundamental pieces in different Django APIs such as models and querysets.

### **30. Why is permanent redirection not a good option?**

Permanent redirection is used only when you don’t want to lead visitors to the old URLs. The response of the permanent redirections is cached by the browser so when you try to redirect to something else it will cause issues. Since this is a browser-side operation if your user wants to move to a new page it will load the same page.

### **31. Difference between Django OneToOneField and ForeignKey Field?**

Both of them are of the most common types of fields used in Django. The only difference between these two is that ForeignKey field consists of on\_delete option along with a model’s class because it’s used for many-to-one relationships while on the other hand, the OneToOneField, only carries out a one-to-one relationship and requires only the model’s class.

### **32. How can you combine multiple QuerySets in a View?**

Initially, Concatenating QuerySets into lists is believed to be the easiest approach. Here’s an example of how to do that:  
from itertools import chain  
result\_list = list(chain(model1\_list, model2\_list, model3\_list))

### **33. How to get a particular item in the Model?**

ModelName.objects.get(id=”term”)  
Note: If there are no results that match the query, get() will raise a ****DoesNotExist**** exception. If more than one item matches the given get() query. In this case, it’ll raise ****MultipleObjectsReturned****, which is also an attribute of the model class itself.

### **34. How to obtain the SQL query from the queryset?**

print(queryset.query)

### **35. What are the ways to customize the functionality of the Django admin interface?**

There are multiple ways to customize the functionality of the Django admin interface. You can piggyback on top of an add/change form that’s automatically generated by Django, you can add JavaScript modules using the js parameter. This parameter is basically a list of URLs that point to the JavaScript modules that are to be included in your project within a <script> tag. You can also write views for the admin if you want.

### **36. Difference between select\_related and prefetch\_related?**

Though both the functions are used to fetch the related fields on a model but their functioning is bit different from each other. In simple words, select\_related uses a foreign key relationship, i.e. using join on the query itself while on the prefetch\_related there is a separate lookup and the joining on the python side. Let’s try to illustrate this via an example:

**from** django.db **import** models**class** **Country**(models.Model):

country\_name = models.CharField(max\_length=5)**class** **State**(models.Model):

state\_name = models.CharField(max\_length=5)

country = model.ForeignKey(Country)

>> states = State.objects.select\_related(‘country’).all()

>> **for** state **in** states:

… print(state.state\_name)

```Query Executed

SELECT state\_id, state\_name, country\_name FROM State INNER JOIN Country ON (State.country\_id = Country.id)

```

>> country = Country.objects.prefetch\_related(‘state’).get(id=1)

>> **for** state **in** country.state.all():

… print(state.state\_name)

```Query Executed

SELECT id, country\_name FROM country WHERE id=1;

SELECT state\_id, state\_name WHERE State WHERE country\_id IN (1);

```

### **37. Explain Q objects in Django ORM?**

Q objects are used to write complex queries, as in filter() functions just `AND` the conditions while if you want to `OR` the conditions you can use Q objects. Let’s see an example:

**from** django.db **import** models**from** django.db.models **import** Q

>> objects = Models.objects.get(

Q(tag\_\_startswith='Human'),

Q(category=’Eyes’) | Q(category=’Nose’)

)

```Query Executed

SELECT \* FROM Model WHERE tag LIKE ‘Human%’ AND (category=’Eyes’ OR category=’Nose’)

```

### **38. What are Django exceptions?**

In addition to the standard Python exceptions, Django raises of its own exceptions.List of the exceptions by Django (https://docs.djangoproject.com/en/3.1/ref/exceptions/)

## ****Q1. What is the difference between Flask and Django?****

|  |  |  |
| --- | --- | --- |
| Comparison Factor | Django | Flask |
| Project Type | Supports large projects | Built for smaller projects |
| Templates, Admin and ORM | Built-in | Requires installation |
| Ease of Learning | Requires more learning and practice | Easy to learn |
| Flexibility | Allows complete web development without the need for third-party tools | More flexible as the user can select any third-party tools according to their choice and requirements |
| Visual Debugging | Does not support Visual Debug | Supports Visual Debug |
| Type of framework | Batteries included | Simple, lightweight |
| Bootstrapping-tool | Built-it | Not available |

## ****Q2. What is Django?****

Django is a **[web development framework](https://www.edureka.co/blog/python-libraries/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank)** that was developed in a fast-paced newsroom. It is a free and open-source framework that was  named after Django Reinhardt who was a jazz guitarist from the 1930s. Django is maintained by a non-profit organization called the Django Software Foundation. The main goal of Django is to enable Web Development quickly and with ease.

## ****Q3. Name some companies that make use of Django?****

Some of the companies that make use of ****Django**** are [Instagram](https://instagram.com/edureka_learning/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank), DISCUS, Mozilla Firefox, YouTube, Pinterest, Reddit, etc.

## ****Q4. What are the features of Django?****

* SEO Optimized
* Extremely fast
* Fully loaded [framework](https://www.edureka.co/blog/python-frameworks/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) that comes along with authentications, content administrations, RSS feeds, etc
* Very secure thereby helping developers avoid common security mistakes such as cross-site request forgery (csrf), clickjacking, cross-site scripting, etc
* It is exceptionally scalable which in turn helps meet the heaviest traffic demands
* Immensely versatile which allows you to develop any kind of websites

## ****Q5. How do you check for the version of Django installed on your system?****

To check for the version of[Django installed on your system](https://docs.djangoproject.com/en/3.1/topics/install/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank), you can open the command prompt and enter the following command:

* python -m django –version

You can also try to import Django and use the get\_version() method as follows:

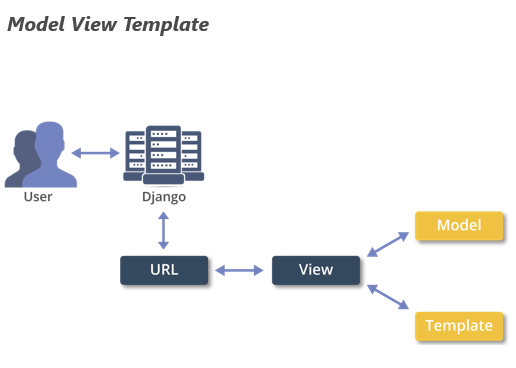
|  |  |
| --- | --- |
| 1  2 | **import** django  print(django.get\_version()) |

## ****Q6. What are the advantages of using Django?****

* Django’s stack is loosely coupled with tight cohesion
* The Django apps make use of very less code
* Allows quick development of websites
* Follows the DRY or the Don’t Repeat Yourself Principle which means, one concept or a piece of data should live in just one place
* Consistent at low as well as high levels
* Behaviors are not implicitly assumed, they are rather explicitly specified
* [SQL statements](https://www.edureka.co/blog/what-is-sql/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) are not executed too many times and are optimized internally
* Can easily drop into raw SQL whenever required
* Flexibility while using URL’s

## ****Q7. Explain Django architecture.****

Django follows the MVT or [Model View Template architecture](https://www.edureka.co/blog/django-tutorial/" \l "architecture" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) whcih is based on the MVC or Model View Controller architecture. The main difference between these two is that Django itself takes care of the controller part.



According to Django, the ‘view’ basically describes the data presented to the user. It does not deal with how the data looks but rather what the data actually is. Views are basically callback functions for the specified URL’s and these callback functions describe which data is presented.

The ‘templates’ on the other hand deal with the presentation of data, thereby, separating the content from its presentation. In Django, views delegate to the templates to present the data.

The ‘controller’ here is Django itself which sends the request to the appropriate view in accordance with the specified URL. This is why Django is referred to as MTV rather than MVC architecture.

## ****Q8. Give a brief about ‘django-admin’.****

django-admin is the command-line utility of Django for administrative tasks. Using the django-admin you can perform a number of tasks some of which are listed out in the following table:

|  |  |
| --- | --- |
| Task | Command |
| To display the usage information and the list of the commands provided by each application | django-admin help |
| To display the list of available commands | django-admin help –command |
| To display the description of a given command and the list of its available options | django-admin help <command> |
| Determining the version of Django | django-admin version |
| Creating new migrations based on the changes made in models | django-admin makemigrations |
| Synchronizing the database state with the current set of models and migrations | django-admin migrate |
| Starting the development server | django-admin runserver |
| Sending a test email in order to confirm the email sending through Django is working | django-admin sendtestemail |
| To start the Python interactive interpreter | django-admin shell |
| To show all the migrations in your project | django-admin showmigrations |

## ****Q9. How do you connect your Django project to the database?****

Django comes with a default [database](https://www.edureka.co/blog/category/databases/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) which is SQLite. To connect your project to this database, use the following commands:

1. python manage.py migrate (migrate command looks at the INSTALLED\_APPS settings and creates database tables accordingly)
2. python manage.py makemigrations (tells Django you have created/ changed your models)
3. python manage.py sqlmigrate <name of the app followed by the generated id> (sqlmigrate takes the migration names and returns their SQL)

## ****Q10. What are the various files that are created when you create a Django Project? Explain briefly.****

When you create a project using the startproject command, the following files will be created:

|  |  |
| --- | --- |
| File Name | Description |
| manage.py | A command-line utility that allows you to interact with your Django project |
| \_\_init\_\_.py | An empty file that tells Python that the current directory should be considered as a Python package |
| settings.py | Consists of the settings for the current project |
| urls.py | Contains the URL’s for the current project |
| wsgi.py | This is an entry-point for the web servers to serve the project you have created |

## ****Django Interview Questions****

## ****Q11. What are ‘Models’?****

Models are a single and definitive source for information about your data. It consists of all the essential fields and behaviors of the data you have stored. Often, each model will map to a single specific database table.

In Django, models serve as the abstraction layer that is used for structuring and manipulating your data. Django models are a subclass of the django.db.models.Model class and the attributes in the models represent database fields.

## ****Q12. What are ‘views’?****

Django views serve the purpose of encapsulation. They encapsulate the logic liable for processing a user’s request and for returning  
the response back to the user. Views in Django either return an HttpResponse or raise an exception such as Http404. HttpResponse contains the objects that consist of the content that is to be rendered to the user. Views can also be used to perform tasks such as read records from the database, delegate to the templates, generate a PDF file, etc.

## ****Q13. What are ‘templates’?****

Django’s template layer renders the information to be presented to the user in a designer-friendly format. Using templates, you can generate [HTML](https://www.edureka.co/blog/what-is-html/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) dynamically. The HTML consists of both static as well as dynamic parts of the content. You can have any number of templates depending on the requirement of your project. It is also fine to have none of them.

[](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank)

### **[Python Django Certification Training Course](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank)**

[Explore Curriculum](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank)

Django has its own template system called the Django template language (DTL). Regardless of the backend, you can also load and render templates using Django’s standard admin.

## ****Q14. What is the difference between a Project and an App?****

An app is basically a Web Application that is created to do something for example, a database of employee records. A project, on the other hand, is a collection of apps of some particular website. Therefore, a single project can consist of ‘n’ number of apps and a single app can be in multiple projects.

## ****Q15. What are the different inheritance styles in Django?****

Django has three possible inheritance styles:

|  |  |
| --- | --- |
| Inheritance style | Description |
| Abstract base classes | Used when you want to use the parent class to hold information that you don’t want to type for each child model. Here, the parent class is never used in solitude |
| Multi-table inheritance | Used when you have to subclass an existing model and want each  model to have its own database table |
| Proxy models | Used if you only want to modify the Python-level behavior of a model, without changing the ‘models’ fields in any way |

## ****Q16. What are static files?****

Static files in Django are those files that serve the purpose of additional files such as the [CSS](https://www.edureka.co/blog/what-is-css/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank), images or [JavaScript](https://www.edureka.co/blog/what-is-javascript/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) files. These files are managed by django.contrib.staticfiles. These files are created within the project app directory by creating a subdirectory named as static.

## ****Q17. What are ‘signals’?****

Django consists of a signal dispatcher that helps allow decoupled applications to get notified when actions occur elsewhere in the framework. Django provides a set of built-in signals that basically allow senders to notify a set of receivers when some action is executed. Some of the signals are as follows:

|  |  |
| --- | --- |
| Signal | Description |
| django.db.models.signals.pre\_save  django.db.models.signals.post\_save | Sent before or after a model’s save() method is called |
| django.db.models.signals.pre\_delete  django.db.models.signals.post\_delete | Sent before or after a model’s delete() method or queryset’s delete() method is called |
| django.db.models.signals.m2m\_changed | Sent when Django starts or finishes an HTTP request |

## ****Q18. Briefly explain Django Field Class.****

‘Field’ is basically an abstract class that actually represents a column in the database table. The Field class, is in turn, a subclass of  RegisterLookupMixin. In Django, these fields are used to create database tables (db\_type()) which are used to map Python types to the database using get\_prep\_value() and vice versa using from\_db\_value() method. Therefore, fields are fundamental pieces in different Django APIs such as models and querysets.

## ****Q19. How to do you create a Django project?****

To create a Django project, cd into the directory where you would like to create your project and type the following command:

* django-admin startproject xyz

****NOTE:****Here, xyz is the name of the project. You can give any name that you desire.

## ****Django Interview Questions****

## ****Q20. What is mixin?****

Mixin is a type of multiple inheritance wherein you can combine behaviors and attributes of more than one parent class. Mixins provide an excellent way to reuse code from multiple classes. For example, generic class-based views consist of a mixin called TemplateResponseMixin whose purpose is to define render\_to\_response() method. When this is combined with a class present in the View, the result will be a TemplateView class.

One drawback of using these mixins is that it becomes difficult to analyze what a child class is doing and which methods to override in case of its code being too scattered between multiple classes.

## ****Q21. What are ‘sessions’?****

Sessions are fully supported in Django. Using the session framework, you can easily store and retrieve arbitrary data based on the per-site-visitors. This framework basically stores data on the server-side and takes care of sending and receiving cookies. These cookies consist of a session ID but not the actual data itself unless you explicitly use a cookie-based backend.

## ****Q22. What do you mean by context?****

Context in Django is a dictionary mapping template variable name given to [Python objects](https://www.edureka.co/blog/python-class/" \l "Objects" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank). This is the conventional name, but you can give any other name of your choice if you wish to do it.

## ****Q23. When can you use iterators in Django ORM?****

Iterators in Python are basically containers that consist of a countable number of elements. Any object that is an iterator implements two methods which are, the \_\_init\_\_() and the \_\_next\_\_()  methods. When you are making use of iterators in Django, the best situation to do it is when you have to process results that will require a large amount of memory space. To do this, you can make use of the iterator() method which basically evaluates a QuerySet and returns the corresponding iterator over the results.

## ****Q24. Explain the caching strategies of Django?****

Caching basically means to save the output of an expensive calculation in order to avoid performing the same calculation again. [Django](https://www.edureka.co/blog/django-tutorial/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) provides a robust cache system which in turn helps you save dynamic web pages so that they don’t have to be evaluated over and over again for each request. Some of the caching strategies of Django are listed down in the following table:

|  |  |
| --- | --- |
| Strategy | Description |
| Memcached | Memory-based cache server which is the fastest and most efficient |
| Filesystem caching | Cache values are stored as separate files in a serialized order |
| Local-memory caching | This is actually the default cache in case you have not specified any other. This type of cache is per-process and thread-safe as well |
| Database caching | Cache data will be stored in the database and works very well if you have a fast and well-indexed database server |

## ****Q25. Explain the use of Middlewares in Django.****

You may come across numerous Django Interview Questions, where you will find this question. Middleware is a framework that is light and low-level plugin system for altering Django’s input and output globally. It is basically a framework of hooks into the request/ response processing of Django. Each component in middleware has some particular task. For example, the AuthenticationMiddleware is used to associate users with requests using sessions. Django provides many other middlewares such as cache middleware to enable site-wide cache, common middleware that performs many tasks such as forbidding access to user agents, URL rewriting, etc, GZip middleware which is used to compress the content for browsers, etc.

## ****Q26. What is the significance of manage.py file in Django?****

The manage.py file is automatically generated whenever you create a project. This is basically a command-line utility that helps you to interact with your Django project in various ways. It does the same things as django-admin but along with that, it also sets the DJANGO\_SETTINGS\_MODULE environment variable in order to point to your project’s settings. Usually, it is better to make use of manage.py rather than the django-admin in case you are working on a single project.

## ****Q27. Explain the use of ‘migrate’ command in Django?****

In Django, migrations are used to propagate changes made to the models. The migrate command is basically used to apply or unapply migrations changes made to the models. This command basically synchronizes the current set of models and migrations with the database state. You can use this command with or without parameters. In case you do not specify any parameter, all apps will have all their migrations running.

## ****Q28. How to view and filter items from the database?****

In order to view all the items from your database, you can make use of the ‘all()’ function in your interactive shell as follows:

* XYZ.objects.all()     where XYZ is some class that you have created in your models

To filter out some element from your database, you either use the get() method or the filter method as follows:

* XYZ.objects.filter(pk=1)
* XYZ.objects.get(id=1)

## ****Q29. Explain how a request is processed in Django?****

In case some user requests a page from some Django powered site, the system follows an algorithm that determines which Python code needs to be executed. Here are the steps that sum up the algorithm:

## ****Django Interview Questions****

## ****Q30. How did Django come into existence?****

Django basically grew from a very practical need. World Online developers namely Adrian Holovaty and Simon Willison started using [Python](https://www.edureka.co/blog/python-basics/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) to develop its websites. As they went on building intensive, richly interactive sites, they began to pull out a generic Web development framework that allowed them to build Web applications more and more quickly. In summer 2005, World Online decided to open-source the resulting software, which is, Django.

## ****Q31. How to use file-based sessions?****

In order to make use of file-based sessions, you will need to set the SESSION\_ENGINE setting to “django.contrib.sessions.backends.  
file”.

## ****Q32. Explain the Django URLs in brief?****

Django allows you to design your own URLs however you like. The aim is to maintain a clean URL scheme without any framework limitations. In order to create URLs for your app, you will need to create a Python module informally called the URLconf or URL configuration which is pure Python code and is also a mapping between the URL path expressions to the Python methods. The length of this mapping can be as long or short as required and can also reference other mappings. When processing a request, the requested URL is matched with the URLs present in the urls.py file and the corresponding view is retrieved. For more details about this, you can refer to the answer to Q29.

## ****Q33. Give the exception classes present in Django.****

Django uses its own exceptions as well as those present in Python. Django core exceptions are present in django.core.exceptions class some of which are mentioned in the table below:

|  |  |
| --- | --- |
| Exception | Description |
| AppRegistryNotReady | Raised when you try to use your models before the app loading process (initializes the ORM) is completed. |
| ObjectDoesNotExist | This is the base class for DoesNotExist exceptions |
| EmptyResultSet | This exception may be raised if a query won’t return any result |
| FieldDoesNotExist | This exception is raised by a model’s \_meta.get\_field() function in case the requested field does not exist |
| MultipleObjectsReturned | This is raised by a query if multiple objects are returned and only one object was expected |

## ****Q34. Is Django stable?****

Yes, Django is quite stable. Many companies like [Instagram](https://instagram.com/edureka_learning/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank), Discus, Pinterest, and Mozilla have been using Django for a duration of many years now. Not just this, Websites that are built using Django have weathered trafﬁc spikes of over 50 thousand hits per second.

## ****Django Interview Questions****

## ****Q35. Does the Django framework scale?****

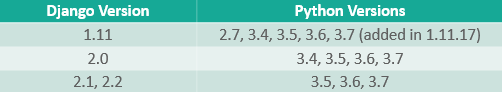
Yes. Hardware is much cheaper when compared to the development time and this is why Django is designed to make full use of any amount of hardware that you can provide it. Django makes use of a “shared-nothing” architecture meaning you can add hardware at any level i.e database servers, caching servers or Web/ application servers.

## ****Q36. Is Django a CMS?****

Django is not a CMS (content-management-system) . It is just a Web framework, a tool that allows you to build websites.

## ****Q37. What Python version should be used with Django?****

The following table gives you the details of the versions of Python that you can use for Django:



Python 3 is actually the most recommended because it is fast, has more features and is better supported. In the case of Python 2.7, Django 1.1 can be used along with it but only till the year 2020.

## ****Q38. Does Django support NoSQL?****

[NoSQL](https://www.edureka.co/blog/sql-vs-nosql-db/" \l "What is NoSQL?" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) basically stands for “not only SQL”. This is considered as an alternative to the traditional RDBMS or the relational Databases.  Officially, Django does not support NoSQL databases. However, there are third-party projects, such as Django non-rel, that allow NoSQL functionality in Django. Currently, you can use MongoDB and Google App Engine.

## ****Q39. How can you customize the functionality of the Django admin interface?****

There are a number of ways to do this. You can piggyback on top of an add/ change form that is automatically generated by Django, you can add JavaScript modules using the js parameter. This parameter is basically a list of URLs that point to the JavaScript modules that are to be included in your project within a <script> tag. In case you want to do more rather than just playing around with from, you can exclusively write views for the admin.

## ****Q40. Is Django better than Flask?****

Django is a framework that allows you to build large projects. On the other hand, [Flask](https://www.youtube.com/watch?v=lj4I_CvBnt0" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) is used to build smaller websites but flask is much easier to learn and use compared to Django. Django is a full-fledged framework and no third-party packages are required. Flask is more of a lightweight framework that allows you to install third-party tools as and how you like. So, the answer to this question basically depends on the user’s need and in case the need is very heavy, the answer is definitely, Django.

## ****Django Interview Questions****

## ****Q41. Give an example of a Django view.****

A view in Django either returns an HttpResponse or raises an exception such as Http404. HttpResponse contains the objects that consist of the content that is to be rendered to the user.

****EXAMPLE:****

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | from django.http import HttpResponse  def hello\_world(request):      html = "  <**h1**>Hello World!</**h1**>    "      return HttpResponse(html) |

****Q42. What should be done in case you get a message saying “Please enter the correct username and password” even after entering the right details to log in to the admin section?****

In case you have entered the right details and still not able to login to the admin site, cross verify if the user account has is\_active and is\_staff attributes set to True. The admin site allows only those users for whom these values are set to True.

## ****Q43. What should be done in case you are not able to log in even after entering the right details and you get no error message?****

In this case, the login cookie is not being set rightly. This happens if the domain of the cookie sent out by Django does not match the domain in your browser. For this, you must change the SESSION\_COOKIE\_DOMAIN setting to match that of your browser.

## ****Q44. How can you limit admin access so that the objects can only be edited by those users who have created them?****

Django’s ModelAdmin class provides customization hooks using which, you can control the visibility and editability of objects in the admin. To do this, you can use the get\_queryset() and has\_change\_permission().

## ****Q45. What to do when you don’t see all objects appearing on the admin site?****

Inconsistent row counts are a result of missing Foreign Key values or if the Foreign Key field is set to null=False. If the ForeignKey points to a record that does not exist and if that foreign is present in the list\_display method, the record will not be shown the admin changelist.

[](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=batches" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank)

### **[Python Django Certification Training Course](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=batches" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank)**

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=batches" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank)

## ****Q46. What do you mean by the csrf\_token?****

The csrf\_token is used for protection against Cross-Site Request Forgeries. This kind of attack takes place when a malicious website consists of a link, some [JavaScript](https://www.edureka.co/blog/javascript-tutorial/" \t "https://www.edureka.co/blog/interview-questions/django-interview-questions/_blank) or a form whose aim is to perform some action on your website by using the login credentials of a genuine user.

## ****Q47. Does Django support multiple-column Primary Keys?****

No. Django only supports single-column Primary Keys.

## ****Q48. How can you see the raw SQL queries that Django is running?****

First, make sure that your DEBUG setting is set to True. Then, type the following commands:

|  |  |
| --- | --- |
| 1  2 | **from** django.db **import** connection  connection.queries |

## ****Q49. Is it mandatory to use the model/ database layer?****

No. The model/ database layer is actually decoupled from the rest of the framework.

## ****Q50. How to make a variable available to all the templates?****

You can make use of the RequestContext in case all your templates require the same objects, such as, in the case of menus. This method takes an HttpRequest as its first parameter and it automatically populates the context with a few variables, according to the engine’s  
context\_processors configuration option.

1. So what is Django?

Django is an open-source web application framework [written in Python](https://www.simplilearn.com/django-interview-questions-article" \o "written in Python" \t "https://www.simplilearn.com/_blank). Developed in a fast-paced newsroom, Django enables the rapid development of easily maintainable and secure websites. It’s a favorite of newbies and advanced programmers alike.

2. Is Django named after that Quentin Tarantino movie?

No, Django is named after Django Reinhardt, a jazz guitarist from the 1930s to the early 1950s who is considered one of the best guitarists of all time.

3. What are Django's most prominent features?

Programmers like Django mostly for its convenient features like:

Optimized for SEO

Extremely fast

A loaded framework that features authentications, content administrations and RSS feeds

Exceptionally scalable to meet the heaviest traffic demand

Highly secure

Versatility, enabling you to create many different types of websites

4. Can you name some companies that use Django?

Some of the more well-known companies that use Django include:’

DISCUS

Instagram

Mozilla Firefox

Pinterest

Reddit

YouTube

5. Why do web developers prefer Django?

Web developers use Django because it:

Allows code modules to be divided into logical groups, making them flexible to change

Provides an auto-generated web admin module to ease website administration

Provides a pre-packaged API for common user tasks

Enables developers to define a given function’s URL

Allows users to separate business logic from the HTML

Is written in Python, one of the most popular[programming languages](https://www.simplilearn.com/best-programming-languages-start-learning-today-article" \o "programming languages" \t "https://www.simplilearn.com/_blank) available today

Gives you a system to define the HTML template for your web page, avoiding code duplication

6. What is CRUD?

It has nothing to do with dirt or grime. It’s a handy acronym for Create, Read, Update, and Delete. It’s a mnemonic framework used to remind developers on how to construct usable models when building application programming interfaces (APIs).

7. Does Django have any drawbacks?

Django’s disadvantages include:

Its monolithic size makes it unsuitable for smaller projects

Everything hinges on Django’s ORM (Object-Relational Mapping)

Everything must be explicitly defined due to a lack of convention

8. What does Django architecture look like?

Django architecture consists of:

Models. Describes the database schema and data structure

Views. Controls what a user sees. The view retrieves data from appropriate models, executes any calculations made, and passes it on to the template

Templates. Controls how the user sees the pages. It describes how the data received from the views need to be altered or formatted to display on the page

Controller. Made up of the Django framework and URL parsing

After going through some of the basic Django interview questions and answers, it is time we increase the difficulty level with the intermediate Django interview questions and answers.

Intermediate Django Interview Questions

Now let’s increase the difficulty factor and explore some tougher Django interview question and answer combinations.

9. In Django’s context, what’s the difference between a project and an app?

The project covers the entire application, while an app is a module or application within the project that deals with one dedicated requirement. So, a project consists of several apps, while an app features in multiple projects.

10. What’s a model in Django?

A model consists of all the necessary fields and attributes of your stored data. They are a single, definitive source of information regarding your data.

11. What are Django’s templates?

Django templates render information in a designer-friendly format to present to the user. Using the Django Template Language (DTL), a user can generate HTML dynamically. Django templates consist of simple text files that can create any text-based format such as XML, CSV, and HTML.

12. Discuss Django’s Request/Response Cycle.

Starting the process off, the Django server receives a request. The server then looks for a matching URL in the URL patterns defined for the project. If the server can’t find a matching URL, it produces a 404-status code. If the URL matches, it executes the corresponding code in the view file associated with the URL and sends a response.

13. What is the Django Admin interface?

Django comes equipped with a fully customizable, built-in admin interface. This portal lets developers see and make changes to all the data residing in the database that contains registered apps and models. The model must be registered in the admin.py file to use a database table with the admin interface.

14. How do you install Django?

Users download and install Python per the operating system used by the host machine. Then run the command pip install “django>=2.2,<3” on the terminal and wait for the installation to finish.

Also Read: [Why Learn Python?](https://www.simplilearn.com/why-learn-python-a-guide-to-unlock-your-python-career-article" \o "Why Learn Python?" \t "https://www.simplilearn.com/_blank)

15. How do you check which version of Django that you have installed on your system?

You can check the version by opening the command prompt and entering the command:

Python-m Django–version

You can also visit the Django homepage [https://www.djangoproject.com/](https://www.djangoproject.com/" \o "https://www.djangoproject.com/" \t "https://www.simplilearn.com/_blank) and look at the “Download latest release” button located on the right of the page.

16. What are signals in Django?

Signals are pieces of code containing information about what is currently going on. A dispatcher is used to both send and listen for signals.

We have covered the easy and intermediate Django interview questions, now let's look into the advanced level of Django interview questions and answers.

Looking forward to a career as a Python Developer? Check out the [Python Training Course](https://www.simplilearn.com/mobile-and-software-development/python-development-training?source=GhPreviewCTAText" \o "Python Training Course" \t "https://www.simplilearn.com/_blank) and get certified today.

Advanced Django Interview Questions

We conclude with eight considerably tougher Django interview questions, designed for the expert-level Django users.

17. What is the Django Rest Framework?

The Django Rest Framework (DRF) is a framework that helps you quickly create RESTful APIs. They are ideal for web applications due to low bandwidth utilization.

18. What do you use middleware for in Django?

You use middleware for four different functions:

Content Gzipping

Cross-site request forgery protection

Session management

Use authentication

19. What does a URLs-config file contain?

The URLs-config file in Django contains a list of URLs and mappings created to view those URLs' functions. The URLs can map to view functions, class-based views, and the URLs-config of other applications.

20. Does Django support multiple-column primary keys?

No, Django supports only single-column primary keys.

21. How can you see raw SQL queries running in Django?

To begin, make sure that the DEBUG setting is set to True. If the setting is squared away, then type the following commands:

1) from Django.db import connection

2) connection.queries

22. List several caching strategies supported by Django.

Django supports these caching strategies:

Database caching

In-memory caching

File System Caching

Memcached

23. What is a QuerySet in the context of Django?

QuerySet is a collection of SQL queries. The command print(b.query) shows you the SQL query created from the Django filter call.

24. What do you use django.test.Client class for?

The Client class acts like a dummy web browser, enabling users to test views and interact with Django-powered applications programmatically. This is especially useful when performing integration testing.

### ****1. So what is Django?****

Django is an open-source web application framework [written in Python](https://www.simplilearn.com/django-interview-questions-article" \o "written in Python" \t "https://www.simplilearn.com/_blank). Developed in a fast-paced newsroom, Django enables the rapid development of easily maintainable and secure websites. It’s a favorite of newbies and advanced programmers alike.

### ****2. Is Django named after that Quentin Tarantino movie?****

No, Django is named after Django Reinhardt, a jazz guitarist from the 1930s to the early 1950s who is considered one of the best guitarists of all time.

### ****3. What are Django's most prominent features?****

Programmers like Django mostly for its convenient features like:

* Optimized for SEO
* Extremely fast
* A loaded framework that features authentications, content administrations and RSS feeds
* Exceptionally scalable to meet the heaviest traffic demand
* Highly secure
* Versatility, enabling you to create many different types of websites

### ****4. Can you name some companies that use Django?****

Some of the more well-known companies that use Django include:’

* DISCUS
* Instagram
* Mozilla Firefox
* Pinterest
* Reddit
* YouTube

### ****5. Why do web developers prefer Django?****

Web developers use Django because it:

* Allows code modules to be divided into logical groups, making them flexible to change
* Provides an auto-generated web admin module to ease website administration
* Provides a pre-packaged API for common user tasks
* Enables developers to define a given function’s URL
* Allows users to separate business logic from the HTML
* Is written in Python, one of the most popular[programming languages](https://www.simplilearn.com/best-programming-languages-start-learning-today-article" \o "programming languages" \t "https://www.simplilearn.com/_blank) available today
* Gives you a system to define the HTML template for your web page, avoiding code duplication

### ****6. What is CRUD?****

It has nothing to do with dirt or grime. It’s a handy acronym for Create, Read, Update, and Delete. It’s a mnemonic framework used to remind developers on how to construct usable models when building application programming interfaces (APIs).

### ****7. Does Django have any drawbacks?****

Django’s disadvantages include:

* Its monolithic size makes it unsuitable for smaller projects
* Everything hinges on Django’s ORM (Object-Relational Mapping)
* Everything must be explicitly defined due to a lack of convention

### ****8. What does Django architecture look like?****

Django architecture consists of:

* **Models**. Describes the database schema and data structure
* **Views.** Controls what a user sees. The view retrieves data from appropriate models, executes any calculations made, and passes it on to the template
* **Templates**. Controls how the user sees the pages. It describes how the data received from the views need to be altered or formatted to display on the page
* **Controller.** Made up of the Django framework and URL parsing

After going through some of the basic Django interview questions and answers, it is time we increase the difficulty level with the intermediate Django interview questions and answers.

## ****Intermediate Django Interview Questions****

Now let’s increase the difficulty factor and explore some tougher Django interview question and answer combinations.

### ****9. In Django’s context, what’s the difference between a project and an app?****

The project covers the entire application, while an app is a module or application within the project that deals with one dedicated requirement. So, a project consists of several apps, while an app features in multiple projects.

### ****10. What’s a model in Django?****

A model consists of all the necessary fields and attributes of your stored data. They are a single, definitive source of information regarding your data.

### ****11. What are Django’s templates?****

Django templates render information in a designer-friendly format to present to the user. Using the Django Template Language (DTL), a user can generate HTML dynamically. Django templates consist of simple text files that can create any text-based format such as XML, CSV, and HTML.

### ****12. Discuss Django’s Request/Response Cycle.****

Starting the process off, the Django server receives a request. The server then looks for a matching URL in the URL patterns defined for the project. If the server can’t find a matching URL, it produces a 404-status code. If the URL matches, it executes the corresponding code in the view file associated with the URL and sends a response.

### ****13. What is the Django Admin interface?****

Django comes equipped with a fully customizable, built-in admin interface. This portal lets developers see and make changes to all the data residing in the database that contains registered apps and models. The model must be registered in the admin.py file to use a database table with the admin interface.

### ****14. How do you install Django?****

Users download and install Python per the operating system used by the host machine. Then run the command pip install “django>=2.2,<3” on the terminal and wait for the installation to finish.

### ****15. How do you check which version of Django that you have installed on your system?****

You can check the version by opening the command prompt and entering the command:

Python-m Django–version

You can also visit the Django homepage [https://www.djangoproject.com/](https://www.djangoproject.com/" \o "https://www.djangoproject.com/" \t "https://www.simplilearn.com/_blank) and look at the “Download latest release” button located on the right of the page.

### ****16. What are signals in Django?****

Signals are pieces of code containing information about what is currently going on. A dispatcher is used to both send and listen for signals.

We have covered the easy and intermediate Django interview questions, now let's look into the advanced level of Django interview questions and answers.

**Advanced Django Interview Questions**

We conclude with eight considerably tougher Django interview questions, designed for the expert-level Django users.

### ****17. What is the Django Rest Framework?****

The Django Rest Framework (DRF) is a framework that helps you quickly create RESTful APIs. They are ideal for web applications due to low bandwidth utilization.

### ****18. What do you use middleware for in Django?****

You use middleware for four different functions:

* Content Gzipping
* Cross-site request forgery protection
* Session management
* Use authentication

### ****19. What does a URLs-config file contain?****

The URLs-config file in Django contains a list of URLs and mappings created to view those URLs' functions. The URLs can map to view functions, class-based views, and the URLs-config of other applications.

### ****20. Does Django support multiple-column primary keys?****

No, Django supports only single-column primary keys.

### ****21. How can you see raw SQL queries running in Django?****

To begin, make sure that the DEBUG setting is set to True. If the setting is squared away, then type the following commands:

1) from Django.db import connection

2) connection.queries

### ****22. List several caching strategies supported by Django.****

Django supports these caching strategies:

* Database caching
* In-memory caching
* File System Caching
* Memcached

### ****23. What is a QuerySet in the context of Django?****

QuerySet is a collection of SQL queries. The command print(b.query) shows you the SQL query created from the Django filter call.

### ****24. What do you use django.test.Client class for?****

The Client class acts like a dummy web browser, enabling users to test views and interact with Django-powered applications programmatically. This is especially useful when performing integration testing.