Top 50 Django Interview Questions and Answers

Q1. What is the difference between Flask and Django?

Flask and Django are both popular web frameworks for Python, but they have different design philosophies and features. Here are the key differences between Flask and Django:

1. Purpose and Complexity:

- **Flask:** Flask is a microframework, which means it provides a minimalistic approach to web development. It offers the basic tools needed to build a web application but leaves many decisions to the developer. Flask provides flexibility and allows you to choose the libraries and components you want to use, making it ideal for small to medium-sized projects and developers who prefer more control.

- **Django:** Django is a full-featured framework that follows the "batteries included" philosophy. It offers a robust set of built-in features, including an ORM (Object-Relational Mapping) for database interaction, an admin interface, authentication, form handling, and more. Django is designed for larger, complex projects and emphasizes convention over configuration, providing a structured and opinionated approach to development.

2. Architecture:

- **Flask:** Flask follows a lightweight and modular design. It does not impose a specific directory structure, allowing developers to organize their code in the way they prefer. Flask provides the core functionalities for handling HTTP requests and responses, URL routing, and templating. Additional features can be added through Flask extensions.

- **Django:** Django follows the Model-View-Controller (MVC) architectural pattern, but with slight variations. It includes an Object-Relational Mapping (ORM) layer called Django ORM, which allows you to define database models as Python classes. Django also provides a templating engine, an admin interface for managing content, and a URL dispatcher for routing requests. It encourages a specific project structure and follows a "convention over configuration" approach.

3. Database Support:

- **Flask:** Flask does not provide an ORM by default. Developers are free to choose any database library or ORM they prefer, such as SQLAlchemy or Django ORM, and integrate it into their Flask application.

- **Django:** Django includes its own powerful ORM, which provides a high-level abstraction for interacting with databases. It supports multiple database backends out of the box, including popular ones like PostgreSQL, MySQL, and SQLite. Django ORM simplifies database operations and helps with tasks like querying, migrations, and data modeling.

4. Community and Ecosystem:

- **Flask:** Flask has a smaller and more flexible ecosystem compared to Django. It has a lightweight core and a wide range of third-party extensions available through the Flask ecosystem. Flask's flexibility allows developers to choose the specific libraries and tools they want to use, which can be advantageous for customizability.

- **Django:** Django has a larger and more established community with extensive documentation and a wide range of built-in features. It offers many pre-built components, which can speed up development time and reduce the need for third-party libraries. Django's ecosystem provides various packages and modules that are specifically built for Django, making it easier to find solutions and resources.

In summary, Flask is a lightweight and flexible framework suitable for smaller projects and developers who prefer more control, while Django is a comprehensive framework with built-in features and conventions designed for larger projects and developers who value productivity and convention. The choice between Flask and Django depends on the project requirements, complexity, and personal preferences of the developers.

Q2. What is Django?

Django is a web development framework 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, DISCUS, Mozilla Firefox, YouTube, Pinterest, Reddit, etc.

Q4. What are the features of Django?

SEO Optimized
Extremely fast
Fully loaded framework 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, 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:

```
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 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 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.

No alt text provided for this image
Model View Template Structure

Acording 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 a command-line tool that comes with the Django web framework. It provides a convenient interface for managing Django projects and performing administrative tasks related to project development and deployment. The `django-admin` tool allows developers to interact with various aspects of a Django project, such as creating new projects, managing database migrations, running development servers, generating Django apps, and more.

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

Django comes with a default database which is SQLite. To connect your project to this database, use the following commands:

python manage.py migrate (migrate command looks at the INSTALLED_APPS settings and creates database tables accordingly)
python manage.py makemigrations (tells Django you have created/ changed your models)
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:

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

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 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.

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:

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, images or JavaScript 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.

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.

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. 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 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:

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 first determines which root URLconf or URL configuration module is to be used
Then, that particular Python module is loaded and then Django looks for the variable urlpatterns
These URL patterns are then run by Django, and it stops at the first match of the requested URL
Once that is done, the Django then imports and calls the given view
In case none of the URLs match the requested URL, Django invokes an error-handling view

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 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:

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, 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.

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:

No alt text provided for this image
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 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 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.

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:

```
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.

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 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:

```
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.