Django Interview Questions

Django is the most popular Python web framework around. It makes it easy to build web apps more quickly and with less code. The demand for Django developers remains high as it's the most sought-after skill set right now.

Rating: 4.5 ★★★★½ 1456

If you're aspiring to become a Django Developer, it's essential to have strong knowledge of these core concepts before appearing for an interview. Through the medium of this article, we are sharing the top 60 most asked Django Interview Questions and Answers that will help you clear the interview with flying colours.

Types of Django Interview Questions

Python Articles

Basic Django Interview Questions and Answers

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

- Python Array Examples
- Python enum
- Python Enumerate with Example
- Python Flask Tutorial For Beginners
- Python For Beginners - Way To Success
- Python For Data Science Tutorial For Beginners

Python Community

- Explore real-time issues getting addressed by experts

Python Quiz

- Test and Explore your knowledge

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.

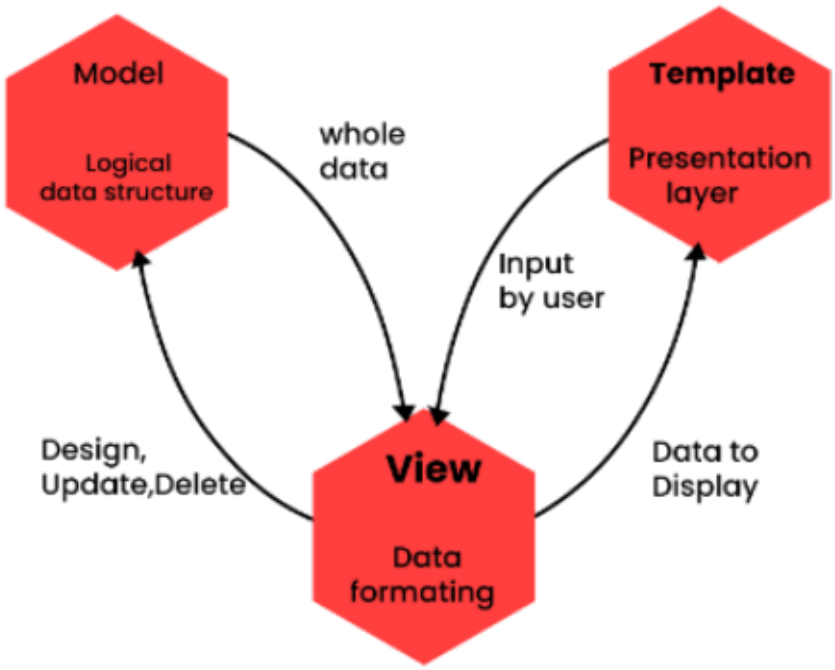
Django is a Python web framework used for full-stack app development and server development.

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:

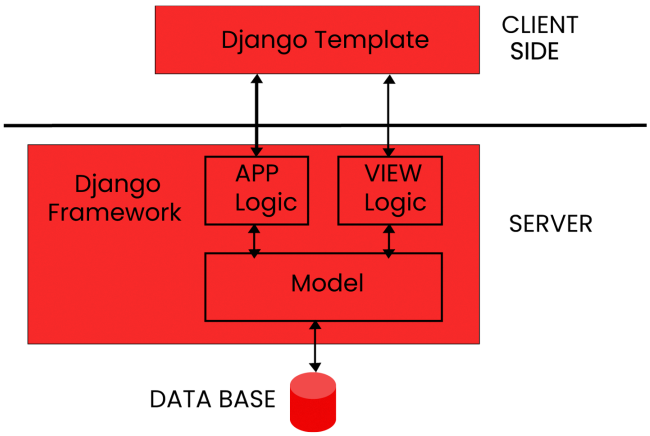
- **Model**: Logical data structure behind the entire app and signified by a database.
- **View:** It's a user interface. What you see when you visit a website is called a user interface. Represented by HTML/CSS/Javascript files.
- **Template:** Deals with the presentation of data.



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 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:

- **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.
- **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.
- **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.

Table of Contents

Signal
Django.db.models.signals.pre_save(or)django.db.models.sign
django.db.models.signals.pre_delete (or)django.db.models.si
django.db.models.signals.m2m_changed
Django.core.signals.request_started(or)django.core.signals.re

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. 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.
	It raises when the user has performed some

SuspiciousOperation	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.
	When Django

ImproperlyConfigured	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, 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
	Displays the usage of the

django-admin help	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** 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
- 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,
```

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 FAQs

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.

Course Schedule

Name	Dates
Python Django Training	Mar 11 to Mar 26

Python Django Training	Mar 14 to Mar 29
Python Django Training	Mar 18 to Apr 02
Python Django Training	Mar 21 to Apr 05

Last updated: 07 March 2023

About Author

Madhuri Yerukala

Madhuri is a Senior Content Creator at MindMajix. She has written about a range of different topics on vario...

Read More

Recommended Courses

Android Training

25 hours 187

4.6

4 /15

EXPLORE COURSES

No comments

Sign up Sign in

Start a conversation ...

Terms of use - Privacy - Report a bug

powered by GraphComment

Trending Courses

Power BI Course | Google Cloud Course | Salesforce Course | Oracle DBA Course | Informatica Course | Snowflake Course | Flutter Course | Python Course | Servicenow Course | Data Science Course | Artificial Intelligence Course | Machine Learning Course

Popular Courses

Agile Course | ArcSight Course | CyberArk Course | Workday Course | Looker Course | AWS Course | Alteryx Course | Powershell Course | Uipath Course | Mulesoft Course | SQl Server DBA Course | React JS Course

Course Categories

Business Intelligence and Analytics Courses | Cloud Computing Courses | Programming and Frameworks Courses | Customer Relationship Management Courses | Database Management & Administration Certification Courses | Business Process Management Courses | Software and Automation Testing Courses | IT Service Management Courses | AI and Machine Learning Courses | ERP Courses

Interview Questions

Salesforce Interview Questions | AWS Interview Questions | RPA Interview Questions | Looker Interview Questions | Informatica Interview Questions | Workday Interview Question | Servicenow Interview Questions |

Flutter Interview Questions | Liferay Interview Questions | Ranorex Interview Questions | Oracle OCI Interview Questions | Citrix Interview Questions | Pega Interview Questions | Tableau Interview Questions | Snowflake Interview Questions | Dart Interview Questions

Tutorials

AWS Tutorial | Salesforce Tutorial | RPA Tutorial | Looker Tutorial | Informatica Tutorial | Workday Tutorial | ServiceNow Tutorial | Power BI Tutorial | SCCM Tutorial | Pega Tutorial | CyberArk Tutorial | Powerapps Tutorial | Workday Tutorial | Netsuite Tutorial | VMware Tutorial | Apache Airflow Tutorial