Top Python Full Stack Developer Interview Questions and Answers

1. What is Django, and what does it use?

Django is a high-level Python web framework used to build secure and scalable applications quickly. It comes with built-in features like authentication, ORM, and admin panels, which speed up development.

2. What is the difference between Django and Flask?

- **Django**: Full-featured framework with batteries-included approach
- **Flask**: Lightweight micro-framework that provides more flexibility
  Django is used for larger projects, while Flask suits smaller or custom architectures.

3. Explain how Django ORM works.

Django ORM allows developers to interact with the database using Python code instead of SQL queries. You define models, and Django automatically translates them into SQL commands.

4. What are REST APIs?

REST APIs allow data communication between frontend and backend systems using HTTP methods like GET, POST, PUT, and DELETE.

5. How do you connect React with a Django backend?

You create REST APIs in Django and consume them in React using tools like fetch() or Axios. Django handles data, while React handles the user interface.

6. What is the purpose of virtualenv?

It creates isolated Python environments so that project dependencies do not conflict with each other.

7. How do you manage static files in Django?

Static files like CSS, JavaScript, and images are handled using Django's STATIC_URL and STATIC_ROOT. During production, they are collected using collectstatic.

8. What is CORS and why is it important?

CORS (Cross-Origin Resource Sharing) controls which domains can make API requests. It is important when React (frontend) and Django (backend) run on different ports.

9. Explain the MVC/MVT architecture in Django.

Django uses MVT:

- **Model**: Database layer
- **View**: Business logic
- **Template**: UI layer
  It separates concerns and improves project structure.

10. What databases are commonly used in Python full stack projects?

MySQL, PostgreSQL, SQLite, and MongoDB are the most common choices depending on project needs.

11. How do you optimize a Django application for performance?

Using caching, indexing, optimized queries, using select_related(), reducing database hits, and implementing pagination.

12. What is JSX in React?

JSX allows writing HTML-like code inside JavaScript. It makes UI development easier and more readable.

13. What is state and props in React?

- **State**: Internal data managed by a component
- **Props**: External data passed from parent to child components

14. Explain Git branching.

Branching lets developers work on features independently without affecting the main code. After finishing, branches are merged back.

15. What is the role of Docker in full stack development?

Docker helps create isolated containers for backend, frontend, and database, making deployment easier and consistent.

16. What are variables in Python?

Variables store values in memory. You don't need to declare the data type separately in Python.

17. What are the basic data types in Python?

Integer, float, string, boolean, list, tuple, dictionary, and set.

18. What is the difference between a list and a tuple?

- List: Mutable (can be changed)
- Tuple: Immutable (cannot be changed)

19. What is a loop?

Loops help repeat a block of code. Python supports for and while loops.

20. What is OOP in Python?

OOP stands for Object-Oriented Programming. It includes concepts like classes, objects, inheritance, and polymorphism.

These basic topics form the foundation of many **python coding interview questions and answers** asked in fresher rounds.

21. What is Django used for?

Django is a Python framework used to build web applications quickly and securely.

22. What are Django models?

Models represent database tables. They allow you to work with data using Python instead of SQL.

23. What is a Django template?

It controls how data is shown to the user in the browser.

24. What is React?

React is a library from JavaScript that is used to build user interfaces.

25. What are the components in React?

Components are reusable pieces of UI, such as buttons, forms, or navigation bars.

26. What is state in React?

State stores data inside a component and can change over time.

These simple Django and React questions help interviewers understand your readiness for full stack work.

27. What are decorators in Python?

Decorators let you change the behavior of a function without modifying the actual code. They are widely used for logging, authentication, and performance tracking.

28. What are generators and why are they useful?

Generators return values one at a time using the yield keyword. They help save memory when working with large datasets.

29. Explain async/await in Python.

async and await support asynchronous programming. They allow your code to run multiple tasks at the same time without blocking the main thread. This is useful in real-time applications like chat apps or dashboards.

30. How do you handle memory optimization in Python?

By using generators, lazy loading, caching, and choosing proper data structures.

These questions fall under typical **advanced python interview questions** asked in senior-level interviews.

31. What is the Django REST Framework (DRF)?

DRF is a toolkit for building APIs quickly. It provides serializers, viewsets, permissions, and authentication support.

32. How do you secure a Django REST API?

- Token or JWT authentication
- Permission classes
- Rate limiting
- Input validation
- HTTPS usage

33. How do you optimize API performance in DRF?

- Using select_related / prefetch_related
- Applying caching
- Reducing database queries
- Using pagination
- Async views in Django 3+

These are common **django rest framework interview questions** that test your API knowledge.

34. How would you design a scalable login system?

Explain:

- Load balancer
- Microservices
- Token-based login
- Database partitioning
- Redis caching
- Rate limiting

35. How do you design a real-time notification system?

Include:

- WebSockets
- Pub/Sub
- Message queues (RabbitMQ, Kafka)
- Background workers

These are typical **system design interview python** topics for senior full stack developers.

36. What are microservices and why use them?

Microservices split an application into smaller services that can be developed and deployed independently. They improve scalability and fault isolation.

37. How do you deploy a Python-based microservice?

Using:

- Docker containers
- Kubernetes

- Nginx/Gunicorn
- CI/CD tools like Jenkins or GitHub Actions

38. Explain CI/CD in a full stack project.

CI/CD automates building, testing, and deployment. It ensures faster releases and fewer production errors.

39. How do you scale a Django application on AWS?

- Use AWS EC2 for servers
- RDS for databases
- Elastic Load Balancer
- Auto Scaling groups
- CloudWatch monitoring
- S3 for media storage

40. Explain Docker in simple terms.

Docker packages your application with all its dependencies into containers that run the same way everywhere.

41. What is Kubernetes used for?

Kubernetes helps manage and scale containers automatically. It handles load distribution, health checks, and deployments.

Common Interview Questions: Python, Django, Flask, and REST API

1. How does Django handle request–response flow?
2. What is the difference between Django and Flask? When would you choose one over the other?
3. Explain serializers and viewsets in Django REST Framework.
4. How do you handle authentication in REST APIs?
5. What are middleware functions in Django and why are they used?

Common Interview Questions: React, HTML, CSS, JavaScript, and API Integration

1. Explain the difference between state and props in React?
2. How do you call a REST API from React?
3. Explain the virtual DOM in simple terms.
4. How do you manage forms and validation in React?
5. What steps do you follow to fix CORS issues when connecting frontend and backend?

Common Interview Questions: SQL, ORM, PostgreSQL

1. Can you explain the difference between SQL and NoSQL databases
2. How do Django ORM queries work internally?
3. What is indexing in a database and why is it important?
4. Explain primary key, foreign key, and unique constraints.
5. How do you optimize a slow SQL query?

Common Interview Questions: Docker, CI/CD, Git, GitHub Pipelines

1. What is Docker and why is containerization important?
2. How do you create a Dockerfile for a Django application?
3. Explain the CI/CD pipeline process in simple terms.
4. What is the role of GitHub Actions in deployment?
5. How do you handle version control best practices in a team?

Common Interview Questions: AWS Basics (Lambda, S3, EC2)

1. How do you store media files in AWS S3 from a Django application?
2. What is AWS Lambda and when should you use serverless functions?
3. How do you deploy a Django project on AWS EC2?

4. Explain the difference between horizontal and vertical scaling.
5. How do you use CloudWatch for monitoring?

Remaining Common Interview Questions:

1. What is CSRF and how does Django prevent it?
2. What is the difference between authentication and authorization?
3. How do you prevent XSS attacks in a web application?
4. What is JWT authentication and how does it work?
5. How do you secure user passwords inside a database?
6. When should you use async programming in a web application?
7. What is the difference between threading vs multiprocessing?
8. How does Python's GIL affect concurrency?
9. How do you implement asynchronous views in Django?
10. How would you scale a Django application serving 1M+ users?
11. What kind of caching strategy would you apply to reduce database load?
12. Explain how a load balancer helps in a full stack application.
13. How do you design a microservice-based system for a large project?
14. When should you use raw SQL instead of ORM?
15. How do you identify slow database queries?
16. What is indexing and how does it improve performance?
17. Explain the N+1 query issue in Django ORM with an example.
18. What is the difference between OAuth2 and JWT?
19. How do you secure REST APIs in Django?
20. What is CSRF and how do you protect against it?

21. How do you implement role-based access control?
22. How do you implement real-time features using Django Channels?
23. What role does Celery play in large systems?
24. How do WebSockets differ from REST APIs?
25. How would you design a task queue for a high-traffic application?