Python Full Stack Interview Q&A;

Q: What are Python's key features that make it suitable for backend development?

A: Easy to learn and read, large standard library, mature frameworks (Django, Flask, FastAPI), strong ecosystem, cross-platform support, and seamless integration with C/C++ for performance.

Q: Explain Python's GIL and its impact on multi-threading.

A: The Global Interpreter Lock (GIL) ensures only one thread executes Python bytecode at a time. It benefits I/O-bound tasks but limits CPU-bound multi-threading. Use multiprocessing to bypass for CPU-heavy work.

Q: Difference between deepcopy and shallow copy?

A: Shallow copy duplicates only the outer object, while deep copy recursively duplicates all nested objects.

Q: What is a generator and how is it different from a regular function?

A: A generator uses 'yield' to return values lazily without storing them all in memory, resuming where it left off.

Q: How does Python handle memory management?

A: Uses reference counting and a cyclic garbage collector. Memory is managed in private heaps.

Q: Compare Django, Flask, and FastAPI.

A: Django: Batteries-included, ORM, auth, admin. Flask: Lightweight, flexible, minimal. FastAPI: Async-first, fast, type hints for validation, smaller ecosystem.

Q: How do you handle authentication in Django?

A: Use 'django.contrib.auth' for user management. For APIs, Django Rest Framework supports session-based or JWT authentication.

Q: What are Django signals?

A: A publish-subscribe pattern allowing decoupled components to listen for events like post_save or pre_delete.

Q: How to implement middleware in Django?

A: Create a class with __init__ and __call__ methods to hook into request/response cycles.

Q: How do you handle AJAX requests in Django?

A: Use JavaScript fetch/axios to send JSON, parse it in Django with json.loads(request.body), return JsonResponse.

Q: Difference between CSR and SSR?

A: CSR (Client-Side Rendering) renders HTML in the browser using JS. SSR (Server-Side Rendering) sends ready HTML from the server.

Q: Difference between select_related and prefetch_related in Django ORM?

A: select_related: JOINs for single-valued relationships. prefetch_related: extra queries for ManyToMany or reverse FK.

Q: How to securely store passwords?

A: Use salted hashes (PBKDF2, bcrypt). Django uses make_password and check_password internally.

Q: How does asyncio work?

A: It uses an event loop to schedule coroutines, enabling cooperative multitasking for I/O-bound tasks.

Q: How to implement WebSockets in Django?

A: Use Django Channels with AsyncWebsocketConsumer to handle connect, receive, disconnect events.

Q: Role of Gunicorn/uWSGI?

A: They are WSGI/ASGI servers that bridge the Python app and web server, handling workers and scaling.

Q: How to manage environment variables securely?

A: Use .env files with python-decouple or os.environ. In production, use secret managers like AWS Secrets Manager.

Q: How to prevent SQL injection in Python web apps?

A: Use ORM query methods or parameterized queries instead of string concatenation.

Q: How to protect against CSRF in Django?

A: Enable CSRF middleware and use {% csrf_token %} in forms. For APIs, use tokens or same-origin policies.

Q: Difference between session-based and token-based authentication?

A: Session-based stores session ID on the server; token-based (JWT) sends a self-contained token with each request.

Q: How to handle caching in a Python web app?

A: Use Redis or Memcached. Django supports caching middleware and per-view caching.

Q: Monolithic vs microservices Python app?

A: Monolith: all features in one codebase, easier to start but harder to scale independently. Microservices: separate deployable services, more scalable but more complex.

Q: Tell me about a challenging bug you fixed.

A: Describe the context, your debugging process, tools used, root cause, and how you prevented it in the future.

Q: How do you handle code reviews and feedback?

A: Be open to feedback, explain reasoning when needed, follow coding standards, and focus on maintainability.