Python Interview Questions (3+ Years Experience)

Q: What are Python’s key features?

A: Python is an interpreted, high-level, dynamically typed, garbage-collected language. It supports multiple paradigms like OOP, functional, and procedural programming. It has rich standard libraries, is highly readable, and integrates well with databases and frameworks.

Q: Explain memory management in Python.

A: Python uses automatic memory management via reference counting and garbage collection (for cyclic references). Memory is managed in private heaps, and developers can use gc module to interact with the garbage collector.

Q: What is the difference between deep copy and shallow copy?

A: A shallow copy copies the object but not nested objects (changes in nested structures reflect in both). A deep copy creates a completely independent clone of the object including nested ones.

Q: Explain @staticmethod, @classmethod, and instance methods.

A: Instance method → operates on object instance (self). Class method → operates on class (cls), can modify class state. Static method → independent of instance & class, utility method.

Q: What are generators and how do they differ from iterators?

A: Generators are functions that yield values lazily using yield. They are a simpler way to build iterators, more memory efficient, and automatically handle state. Iterators, on the other hand, need a class implementing __iter__() and __next__().

Q: What’s the difference between list, tuple, set, and dict?

A: List → ordered, mutable, allows duplicates. Tuple → ordered, immutable. Set → unordered, unique elements. Dict → key-value mapping.

Q: What are Python decorators?

A: A decorator is a function that modifies another function or method without changing its code. They are widely used for logging, authentication, caching, etc.

Q: How does GIL (Global Interpreter Lock) affect multithreading?

A: GIL ensures only one thread executes Python bytecode at a time, which limits CPU-bound multithreading. For I/O-bound tasks, multithreading works fine, but for CPU-bound tasks, multiprocessing is better.

Q: What are some differences between Django, Flask, and FastAPI?

A: Django → full-stack, batteries-included, ORM, admin panel. Flask → lightweight, flexible, minimal. FastAPI → modern, async-first, type-hints support, great for APIs.

Q: Explain REST API principles.

A: REST APIs follow statelessness, resource-based URLs, HTTP methods (GET, POST, PUT, DELETE), and data exchange typically via JSON. They use status codes and should be cacheable and scalable.

Q: How do you connect Python with SQL/NoSQL databases?

A: For SQL: use libraries like psycopg2 (Postgres), mysql-connector, or ORM like Django ORM/SQLAlchemy. For NoSQL: use pymongo for MongoDB, or redis-py for Redis.

Q: What are Python’s data types?

A: Numeric (int, float, complex), Sequence (list, tuple, range), Text (str), Mapping (dict), Set (set, frozenset), Boolean (bool), NoneType.

Q: Explain exception handling in Python.

A: Use try-except-finally blocks. finally always executes. Custom exceptions can be created by extending Exception class.

Q: How do you deploy a Flask project?

A: 1. Create virtual environment 2. Install dependencies (requirements.txt) 3. Run via WSGI server (Gunicorn/uWSGI) 4. Use Nginx/Apache as reverse proxy 5. Containerize with Docker if needed 6. Deploy to AWS/GCP/Azure/Heroku

Q: Scenario: Your API response is taking too long. How do you optimize it?

A: Profile the code (cProfile, line_profiler). Optimize queries (use joins, indexes, ORM optimizations). Use caching (Redis, memcached). Paginate large responses. Use async (FastAPI, asyncio, aiohttp).