- **Question1:** Describe a situation where you leveraged Flask or FastAPI for a micro-service architecture. What were the key challenges and how did you overcome them? **Answer:** In a recent project, I used FastAPI to build a micro-service for user authentication. The key challenges included managing asynchronous operations and ensuring seamless communication between micro-services. I overcame these by implementing proper exception handling, using asynchronous database operations, and integrating with a message broker like RabbitMQ to handle inter-service communication.

- **Question2:** How do you implement concurrency in Python, and what are the advantages of using asyncio compared to traditional threading? **Answer:** Concurrency in Python can be implemented using threading, multiprocessing, or asyncio. Asyncio provides a single-threaded, cooperative multitasking approach, which is advantageous in I/O-bound tasks because it avoids the overhead of thread management and context switching, leading to better performance and scalability for certain applications.

- **Question3:** Can you explain the principles of test-driven development (TDD) and how you apply them in a Python project? **Answer:** TDD involves writing tests before the actual code. The process includes writing a test case for a new function, running the test to see it fail, writing the minimum code to pass the test, and then refactoring the code while keeping the test green. In Python, I use PyTest to write and manage tests, ensuring that each new feature or bug fix is covered by corresponding test cases.

- **Question4:** What are some modern design patterns you have used in your Python projects? Provide examples of how they were implemented. **Answer:** I've used several design patterns, including the Singleton pattern for managing database connections, the Factory pattern for creating objects based on different input conditions, and the Observer pattern for implementing event-driven architectures. For instance, in a Django project, I used the Singleton pattern to ensure only one instance of a database connection pool was created.

- **Question5:** How do you handle RESTful service implementation in Django, and what tools or libraries do you use to enhance this process? **Answer:** In Django, I handle RESTful service implementation using Django REST Framework (DRF). DRF simplifies the creation of RESTful APIs with features like serializers, viewsets, and routers. Additionally, I use tools like Postman for testing APIs and Swagger for auto-generating API documentation.

- **Question6:** Describe a scenario where you used Docker and Kubernetes to deploy a Python application. What were the key steps and considerations? **Answer:** In a recent deployment, I containerized a Python web application using Docker by writing a Dockerfile and creating Docker images. I then used Kubernetes for orchestration, defining deployment and service manifests to manage replicas and load balancing. Key considerations included ensuring statelessness for the application containers, setting up persistent storage for databases, and configuring health checks for automated restarts.

- **Question7:** How do you utilize mocking frameworks in your unit tests? Provide an example of a complex test case that involved mocking. **Answer:** Mocking frameworks like unittest.mock in Python allow me to simulate objects and their behaviors. For example, in a test case for an

API endpoint that interacts with an external payment service, I used mocks to simulate the service's responses. This enabled me to test different scenarios like successful payments, declined transactions, and network errors without making actual API calls.

- **Question8:** What are some challenges you have faced with version control using Git, and how did you address them? **Answer:** Challenges with Git include managing merge conflicts in a large team and maintaining a clean commit history. I addressed merge conflicts by frequent communication with team members, using feature branches, and conducting regular code reviews. To maintain a clean history, I used interactive rebase to squash commits and amend commit messages.

- **Question9:** Explain a project where you utilized GCP technologies like BigQuery or DataFlow. What were the objectives and outcomes? **Answer:** In a data analytics project, I used BigQuery for processing and analyzing large datasets. The objective was to generate insights from user activity logs. I leveraged BigQuery's SQL capabilities to write complex queries and DataFlow for data transformation pipelines. The outcome was a set of interactive dashboards that provided real-time insights into user behavior, which helped in making data-driven decisions.

- **Question10:** How do you ensure your code adheres to modern design principles and is maintainable for future development? **Answer:** To ensure adherence to modern design principles and maintainability, I follow best practices like writing clean and readable code, using design patterns appropriately, adhering to SOLID principles, and maintaining comprehensive documentation. I also enforce coding standards through code reviews, use linters like pylint, and ensure thorough test coverage with unit and integration tests.