**Question 1: Explain how FastAPI handles asynchronous requests and its benefits over synchronous code in Python.**

FastAPI utilizes asynchronous programming to handle requests efficiently. It is built on top of Starlette, an asynchronous web framework. FastAPI allows you to define asynchronous route handlers using async def instead of the traditional def for synchronous functions.

The benefits of using asynchronous code in FastAPI include:

* Improved Performance: Asynchronous code allows handling multiple requests concurrently without waiting for one to finish before moving on to the next. This can significantly improve the overall performance of the application.
* Scalability: Asynchronous code is well-suited for I/O-bound operations, such as making requests to databases or external APIs. It enables the application to scale better and handle more concurrent users.
* Responsive User Interfaces: Asynchronous code is particularly beneficial in scenarios where the application needs to remain responsive, like handling multiple requests simultaneously in web applications.

Example of asynchronous route handler in FastAPI:

from fastapi import FastAPI

app = FastAPI()

@app.get("/async\_example")

async def async\_example():

result = await some\_async\_function()

return {"result": result}

**Question 2: Describe how dependency injection works in FastAPI and give an example of its practical use.**

FastAPI leverages dependency injection to manage and inject dependencies into route handlers or other parts of the application. Dependencies can include things like database connections, authentication checks, or any other resource required by your application.

Here's an example of dependency injection in FastAPI:

from fastapi import Depends, FastAPI, HTTPException

# Dependency

def get\_token\_header(x\_token: str = Depends(lambda x: x.headers["x-token"])):

if x\_token != "fake-super-secret-token":

raise HTTPException(status\_code=400, detail="X-Token header invalid")

return x\_token

app = FastAPI()

@app.get("/dependency\_example/")

async def dependency\_example(token: str = Depends(get\_token\_header)):

return {"token": token}

In this example, the get\_token\_header function is a dependency that extracts the x-token header from the request. It is then injected into the dependency\_example route handler as a parameter. If the header is invalid, FastAPI will automatically raise an HTTPException.