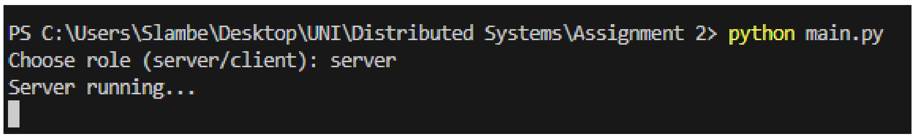
Explanation of the code execution:

1. Server initialization
    - Creating an instance of the 'NoteServer' class initializes the server.
    - SimpleXMLRPCServer is created on localhost port 8000.
        - SimpleXMLRPCServer is a class provided by the Python standard library's xmlrpc.server module. It simplifies the creation of an XML-RPC server in Python by abstracting away low-level networking details.
        - SimpleXMLRPCServer provides methods for registering functions or objects to handle XML-RPC requests, setting up network connections, and serving requests indefinitely.
    - The instance of the server is registered with the XML-RPC server.
        - An XML-RPC server is component that listens for incoming XML-RPC requests from clients over a network.
        - When a request is received, the server processes it, executes the requested method or procedure, and sends back the result to the client.
2. Running the server

- The server runs and waits for client requests (the server is started with simply running the program and choosing server as role:



```
PS C:\Users\Slambe\Desktop\UNI\Distributed Systems\Assignment 2> python main.py
Choose role (server/client): server
Server running...
```

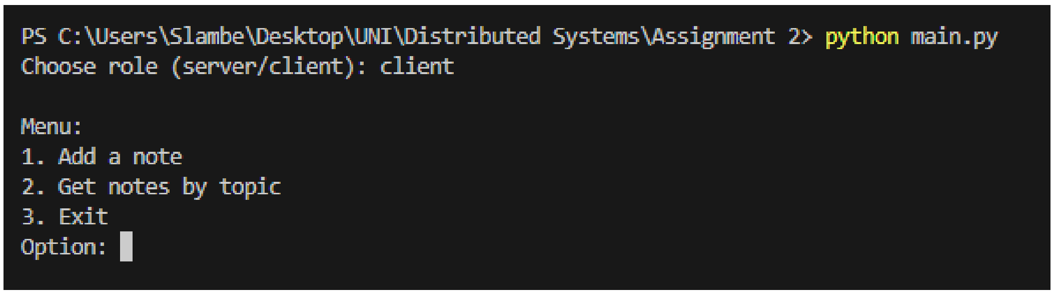
- The server listens for RPC requests and delegates them to the methods in the NoteServer class.

3. Initializing the client

- Creating an instance of the NoteClient class initializes the client.
- The client uses ServerProxy to connect to the server's XML-RPC endpoint.

4. Interacting with clients

- The client prints out a menu to the user.



```
PS C:\Users\Slambe\Desktop\UNI\Distributed Systems\Assignment 2> python main.py
Choose role (server/client): client

Menu:
1. Add a note
2. Get notes by topic
3. Exit
Option:
```

- User's choice determines how the client interacts with the server by making RPC calls.
    - The client sends RPC requests (add_note, get_notes) to the server and receives responses.

Breakdown of Remote Procedure Calls (RPC):

Server side:

- The server's methods (add_note, get_notes) can be remotely invoked by clients.
    - Registered with the XML-RPC server at initialization.

Client Side:

- The client calls methods exposed by the server using XML-RPC ServerProxy.
- The client's method calls are translated into RPC requests and sent over the network to the server.
    - When the server responds, the client method call returns with the result.

Failure Handling

1. If the client fails to connect to the server or experiences a network error, exceptions are raised (try-except blocks in the code).
2. If the server encounters errors during RPC method execution, it raises exceptions that are handled by the client.

Design Challenges of Distributed Systems

1. Heterogeneity
- The system is designed to support components written in different languages and XML-RPC handles interoperability (Python for server and client).
2. Openness
- The RPC interfaces add_note and get_notes are clear and allow new components to be added or removed without breaking the code.
3. Security
- In real-world application security measures should be implemented (authentication, authorization, data encryption…) but they are not present in this task code.
4. Scalability
- The code is meant to handle increasing number of clients by using asynchronous processing which is done by threading in Python.
5. Failure Handling
- As stated earlier, try-catch blocks in the code handle some error cases but in real-world application more robustness should be implemented to this.
6. Transparency
- The distributed system is transparent to clients. Clients interact with the server using familiar calls and the underlying RPC communication is abstracted away.