XML-RPC tutorial

Introduction to XML-RPC

Server in Python

```
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler

# Restrict to a particular path.
class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)

# Create server
with SimpleXMLRPCServer(('localhost', 8000),
                        requestHandler=RequestHandler) as server:
    server.register_introspection_functions()

    # Register pow() function; this will use the value of
    # pow.__name__ as the name, which is just 'pow'.
    server.register_function(pow)

    # Register a function under a different name
    def adder_function(x, y):
        return x + y
    server.register_function(adder_function, 'add')

    # Register an instance; all the methods of the instance are
    # published as XML-RPC methods (in this case, just 'mul').
    class MyFuncs:
        def mul(self, x, y):
            return x * y

    server.register_instance(MyFuncs())

    # Run the server's main loop
    server.serve_forever()
```

Client - in Python

```
import xmlrpc.client
```

```
s = xmlrpc.client.ServerProxy('http://localhost:8000')
print(s.pow(2,3))  # Returns 2**3 = 8
print(s.add(2,3))  # Returns 5
print(s.mul(5,2))  # Returns 5*2 = 10

# Print list of available methods
print(s.system.listMethods())
```

XML-RPC Programming Exercise – Remote Task Manager

The goal of this exercise is to understand how **Remote Procedure Calls (RPC)** work in a client–server architecture using the XML-RPC protocol. Implement a simple **task management system** consisting of a Python **XML-RPC server** and a Python **client application**.

Server

Create an XML-RPC server in Python that manages a list of tasks.

Each task should contain the following fields:

- **id** – unique integer identifier
- **title** – short text describing the task
- **completed** – boolean value indicating whether the task is finished

The server must provide the following remote methods:

Method	**Description**
add_task(title)	Creates a new task and returns the generated task ID
list_tasks()	Returns all tasks as a list of dictionaries
complete_task(task_id)	Marks the specified task as completed
delete_task(task_id)	Removes the specified task
get_task(task_id)	Returns the details of a single task

The server should store tasks **in memory** while it is running.

Client

Write a Python client that connects to the XML-RPC server and allows the user to interact with the task manager.

The client program should:

- connect to the XML-RPC server
- call the remote methods
- display results in a readable format

The client should provide a simple **command line menu**, for example:

```
1 - Add task
2 - List tasks
3 - Complete task
4 - Delete task
5 - Show task details
0 - Exit
```

Data Format

All data exchanged between the client and server must use **XML-RPC compatible data types**, such as:

- integers
- strings
- booleans
- lists
- dictionaries