zlib compression in HTTP Server (Client)

Description

In this problem, you need to implement an HTTP server that supports zlib compression using Python's built-in libraries. The server should handle GET requests to the /status endpoint and return a JSON response. When the client includes **Accept-Encoding: deflate** in the request headers, the server must compress the response using **zlib** compression.

Your task is to create both:

1. An HTTP server that listens on localhost:8080 and handles compression
2. An HTTP client that can connect to the server and decompress responses

Client Requirements:

- Connect to localhost:8080
- Send GET request to /status with Accept-Encoding: deflate
- Receive response in 1024-byte chunks
- Extract headers and body from HTTP response
- Decompress body using zlib when Content-Encoding: deflate is present
- Parse the decompressed JSON data
- Display headers, compressed body bytes, and parsed JSON object

Input

The programs should not require any command-line input. Both server and client should be implemented as standalone Python scripts that can be executed directly.

Output (without unit test)

Client Output:

== HEADERS ==

HTTP/1.1 200 OK

Content-Type: application/json

```
Content-Encoding: deflate

Content-Length: 46


== BODY ==

b'x\x9c\xabV\xcaK\xccMU\xb2RP\xca\xad\x0cN-*K-R\xd2QP*.I,)-
\x06\x89\xe6\xe7\xe5d\xe6\xa5*\xd5\x02\x00\n\xeb\r0'


== JSON OBJECT ==

{'name': 'myServer', 'status': 'online'}
```

Output (with unit test)

When running the unit tests for both server and client implementations, the expected output should show successful test assertions:

Client Test Output:

✅ Client connects to localhost:8080

✅ Client sends correct HTTP request with deflate encoding

✅ Client receives response data in 1024-byte chunks

✅ Client closes connection after receiving response

✅ Client correctly extracts headers from response

✅ Client correctly extracts and decompresses zlib data

✅ Client correctly parses decompressed JSON data