Simple File Download (Client)

Description
In this challenge, you are required to complete and test a client application written in Python. The client connects to a file server, sends a download request for a specific file, receives the file along with its metadata (e.g., file name and size), and saves the file locally. Your task includes implementing the client-side functionality to handle server responses correctly and manage file data reception.

Input
Your client will be given the hostname (or IP address) and port number of the file server, along with the name of the file to be downloaded. The input format to your client application will be:
- Hostname or IP address
- Port number
- Command and filename, e.g., "download example.txt"

Output (when testing to communicate with the server)
The client should output:
- The status of the connection attempt to the server.
- Any response received from the server regarding the command sent (e.g., success or error messages).
- Confirmation of file receipt and its storage location, if applicable.

Constraints
- The client should manage network communication properly, including connecting to and disconnecting from the server.
- The client must correctly parse the metadata header sent by the server.

Example:
For a file named "example.txt" available on the server:
- Client input: "download example.txt"
- Client output:
    Connecting to [server]:[port]
    "{file_name} has been received successfully!"

Expected unit test output

```
test attribute passed: localhost is equal to localhost
test attribute passed: 65432 is equal to 65432
test attribute passed: 127.0.0.1 is equal to 127.0.0.1
test attribute passed: 65432 is equal to 65432
test attribute passed: localhost is equal to localhost
test attribute passed: 5000 is equal to 5000
test attribute passed: 127.0.0.1 is equal to 127.0.0.1
test attribute passed: 5000 is equal to 5000

Testing connect to server ...
Connecting to localhost:65432
connect called with: call(('localhost', 65432))
```

```
Testing disconnect ...
close called with: call()

Testing parse header ...
test attribute passed: example.txt is equal to example.txt
test attribute passed: 1024 is equal to 1024
test attribute passed: test content is equal to test content

Testing receive message ...
recv return value: b'data'
recv called with: call(1024)

Testing send message ...
send called with: call(b'Hello')
```