Simplified IoT Light Bulb

The **client** will perform the following functions:

1. Read in 3 required arguments from the command line, and potentially up to 2 more arguments.
    a. IP Address of Server (`127.0.0.1`)
    b. Port of Server (e.g. `9999`)
    c. Command to be executed (`UPDATE` or `READ`)
    d. Power (`on` or `off`)
    e. (required only if power=on) Color (`red, orange, yellow, green, blue,` or `white`)
2. Send these operation requests, with a timeout of 1 second.
    a. If a response arrives within the timeout period, the status will print to the console.
    b. If not, the request will be re-sent for a maximum of 3 attempts before printing an error message and exiting.

How the arguments are expected to be formatted:

```
serverIP serverPort command [power [color]]
```

The command portion of the command-line arguments follows the format:

```
COMMAND power color
```

An example command may read:

```
UPDATE on red
```

which tells the server we are **updating** the bulb to turn **on** and setting the color to **red**.

Another example command:

```
READ
```

which tells the server we want to **read** the status of the power and color.

Supported commands and their parameters

UPDATE

Update the power and/or color of the bulb.

`power:` `on` or `off`
The power to set the bulb to.

`color:` `red, orange, yellow, green, blue,` or `white`
The color to set the bulb to. Required only if `power` is set to `on`.

READ

Read the power and color of the bulb.

no positional parameters for this command

The **server** will perform the following functions:

1. Read in 2 arguments from the command line
    a. IP Address of Server (`127.0.0.1`)
    b. Port of Server (e.g. 9999)
2. Receive and interpret commands send from the Client.
3. Perform the operations the client sends, and return appropriate status response codes
    a. If all good, respond with "`OK`" and the state of bulb (e.g. "`OK on orange`")
    b. If an error occurs, respond with an error traceback detailing what went wrong

Message Format

With this IoT Lightbulb, network requests have application data formatted in network byte order as follows:

Example Client Request

0	1	2	3
Msg Type (0)		Return Code (0)	
Request Identifier (15)			
Command String Length (6)		Return String Length (2)	
Command String ("UPDATE on red")			

Example Server Response

0	1	2	3
Msg Type (1)		Return Code (0)	
Request Identifier (15)			
Command String Length (6)		Return String Length (2)	
Command String ("UPDATE on red")			
Return String ("OK on red")			

Msg Type (2 bytes):
In Request, use 0.
In Response, use 1.

Return Code (2 bytes):
In Request, use 0 for OK.
In Response, use 0 for OK and 1 for ERROR.

Request Identifier (4 bytes):
In Request, this is a 4B unsigned integer value representing the sequence number of the request.
In Response, echo the value back.

Command String Length (2 bytes):
In Request, this is a 2B unsigned integer value representing the length of the command to-be-performed.
In Response, echo the value back.

Return String Length (2 bytes):
In Request, use 0.
In Response, this is a 2B unsigned integer value representing the length of the string that the server responds with.

Command (variable length):
In Request, this is a variable Length string representing the command and any parameters.

Return String (variable length):
In Request, this field is left empty.
In Response, this field represents status messages. If the request is received and performed well, the server will respond with and the state of bulb (e.g. “OK on orange”). If there is an error, the server will respond with an error traceback.

Error Handling

CommandError:
Raised when the first word of the Command String is not a supported Command.
Return: “`ERROR <COMMAND> is not a supported command`.”

PowerError:
Raised when the power parameter of the UPDATE command is not supported.
Return: “`ERROR <power> is not on/off`”

ColorError:
Raised when the color parameter passed during an UPDATE command is unsupported.
Return string should be formatted as “`ERROR <color> is not a supported color.`” where <color> is the passed argument.

MessageFormatError:
Raised when the string does not follow the format “`COMMAND power color`”.
Return string should be formatted as “`ERROR Message format unrecognized`.

Test Output

Successful Commands

Client Request 1

```
$ python iot-client.py 127.0.0.1 9999 UPDATE on red
Sending request to 127.0.0.1, 9999
ID: 1
Command: UPDATE on red
Received response from 127.0.0.1, 9999
ID: 1
Status: OK on red
```

Client Request 2

```
$ python iot-client.py 127.0.0.1 9999 UPDATE on yellow
Sending request to 127.0.0.1, 9999
ID: 2
Command: UPDATE on yellow
Received response from 127.0.0.1, 9999
ID: 2
Status: OK on yellow
```

Client Request 3

```
$ python iot-client.py 127.0.0.1 9999 READ
Sending request to 127.0.0.1, 9999
ID: 3
Command: READ
Received response from 127.0.0.1, 9999
ID: 3
Status: OK on red
```

Client Request 4

```
$ python iot-client.py 127.0.0.1 9999 UPDATE off
Sending request to 127.0.0.1, 9999
ID: 4
Command: UPDATE off
Received response from 127.0.0.1, 9999
ID: 4
Status: OK off
```

Server (Running for all 4 examples)

```
$ python iot-server.py 127.0.0.1 9999
Receiving requests from 127.0.0.1, 9999...

Received request from 127.0.0.1, 5000
ID: 1
Command: UPDATE on red
Sending response to 127.0.0.1, 5000
ID: 1
Status: OK on red

Received request from 127.0.0.1, 5000
ID: 2
Command: UPDATE on yellow
Sending response to 127.0.0.1, 5000
ID: 2
Status: OK on yellow

Received request from 127.0.0.1, 5000
ID: 3
Command: READ
Sending response to 127.0.0.1, 5000
ID: 3
Status: OK on yellow

Received request from 127.0.0.1, 5000
ID: 4
Command: OK off
Sending response to 127.0.0.1, 5000
ID: 4
Status: OK on red
```

Error Handling

1. Command Error

Client

```
$ python iot-client.py 127.0.0.1 9999 VOLUME down
Sending request to 127.0.0.1, 9999
ID: 5
Command: VOLUME down
Received response from 127.0.0.1, 9999
ID: 5
Status: ERROR VOLUME is not a supported command.
```

Server

```
$ python iot-server.py 127.0.0.1 9999
Receiving requests from 127.0.0.1, 9999...

Received request from 127.0.0.1, 5000
ID: 5
Command: VOLUME down
Sending response to 127.0.0.1, 5000
ID: 5
Status: ERROR VOLUME is not a supported command.
```

2. Color Error

Client

```
$ python iot-client.py 127.0.0.1 9999 UPDATE on magenta
Sending request to 127.0.0.1, 9999
ID: 6
Command: UPDATE on magenta
Received response from 127.0.0.1, 9999
ID: 6
Status: ERROR Unsupported color magenta
```

Server

```
$ python iot-server.py 127.0.0.1 9999
Receiving requests from 127.0.0.1, 9999...

Received request from 127.0.0.1, 5000
ID: 6
Command: UPDATE on magenta
Sending response to 127.0.0.1, 5000
ID: 6
Status: ERROR Unsupported color magenta
```

3. Power Error

```
$ python iot-client.py 127.0.0.1 9999 UPDATE notapower
Sending requests to 127.0.0.1, 9999
ERROR notapower is not on/off.
```

Client

```
$ python iot-client.py 127.0.0.1 9999 UPDATE notapower
Sending request to 127.0.0.1, 9999
ID: 7
Command: UPDATE notapower
Received response from 127.0.0.1, 9999
ID: 7
Status: ERRROR notapower is not on/off
```

Server

```
$ python iot-server.py 127.0.0.1 9999
Receiving requests from 127.0.0.1, 9999...

Received request from 127.0.0.1, 5000
ID: 7
Command: UPDATE notapower
Sending response to 127.0.0.1, 5000
ID: 7
Status: ERROR notapower is not on/off
```

4. Unresponsive server

Client

```
$ python iot-client.py 127.0.0.1 9999 UPDATE on blue
Sending requests to 127.0.0.1, 9999
ID: 0
Command: UPDATE on blue
Request 0 timed out…
Request 0 timed out…
Request 0 timed out…
Server isn’t responding. Exiting
```

Server

server is off