Homework: Python web server
Due Monday, Feb 2
50 points

You must submit your own work for this assignment. You are *not* required to work every detail out on your own; if you need help please ask.

You may hand in printed hard copy, email it to me, or share it to me in github (send me repository and commit information).

Using code you've written, or one of the python servers from the course github, complete the implementation of a basic web server in python.

Requirements:

- Accept client connections in a loop (your server may be single threaded but should handle multiple requests and not exit after the first request)
- parse HTTP requests to isolate the path for use in generating a response
  - this can be accomplished using the string's split() method and/or slices, it doesn't need to be complicated
- establish a doc_root variable which stores a file system path holding your web files
- establish a doc_index variable which stores the default file to serve if the request is for a directory instead of a file
  - i.e. if the client requests a path such as "/", return the file named index.html
  - if the specified file does not exist, make sure to send an appropriate response, e.g. a directory listing with links or an error status and page.
- compose a response message including
  - an appropriate status code
  - include a `connection: close header`
  - a content-type header (at minimum for text, html, jpg, png, gif, css and js files), e.g. `content-type: text/html`
  - the file, if it is found, as content
- additionally, allow for redirection for specific paths with a 302 status code and location header in the response.

Details:
The existing programs use webob to construct HTTP responses. You may use webob, write your own HTTP response class/function if you prefer that, or use workzeug as long as your code is clear and includes remarks if needed.

The server must serve a complete web page to a common web browser, e.g. firefox or chrome. Be sure to test so that you can handle quirks like the request for /favicon.ico with every page.

You may read your redirects from a file or hard code them into your program. Store them as a python dictionary, e.g.

```
redirects = { '/google': 'http://www.google.com/',
              '/bu': 'http://www.butler.edu/' }
```

There are patterns for iterating over keys, values or both keys and values in python dictionaries:

```
# only keys
for key in redirects:
    pass
# only values
for value in redirects.itervalues():
    pass
# keys and values
for key, value in redirects.iteritems():
    pass
```