Michael Price & Shan Lalani
CS560 Programming Assignment 1
server.py Documentation

Overview and Basic Usage

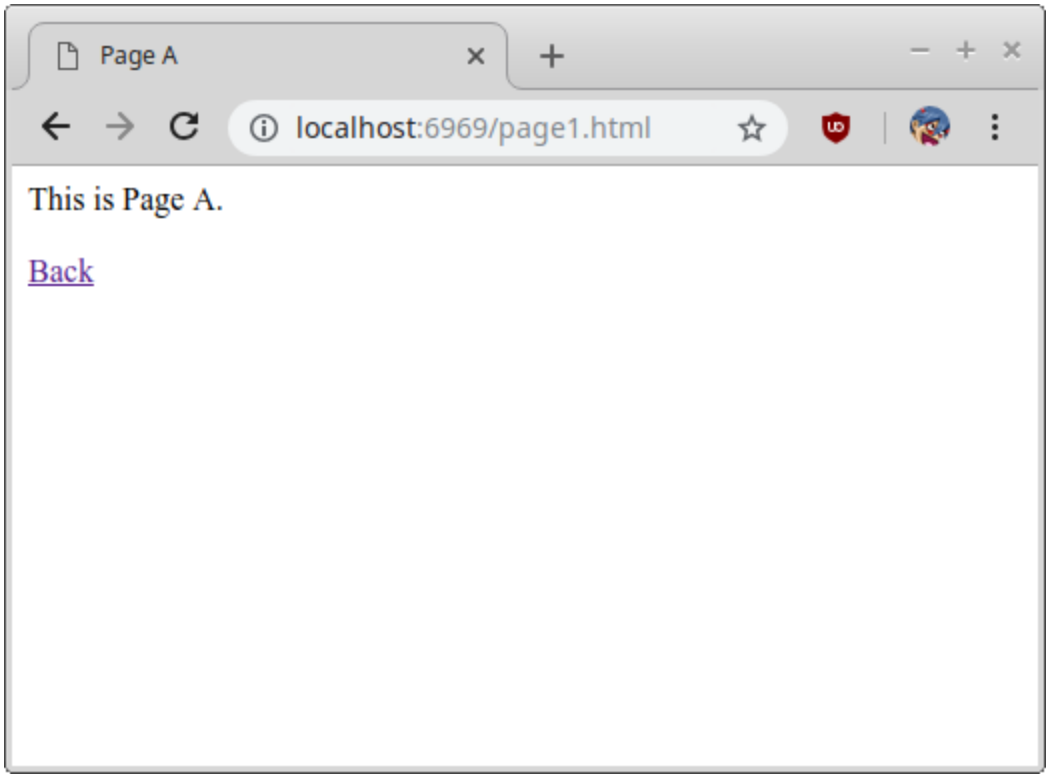
The server is implemented using Python 3, specifically with the socket library. It can be executed on the terminal using the command *python3 server.py* in the folder containing the aforementioned file.

Usage of Sockets

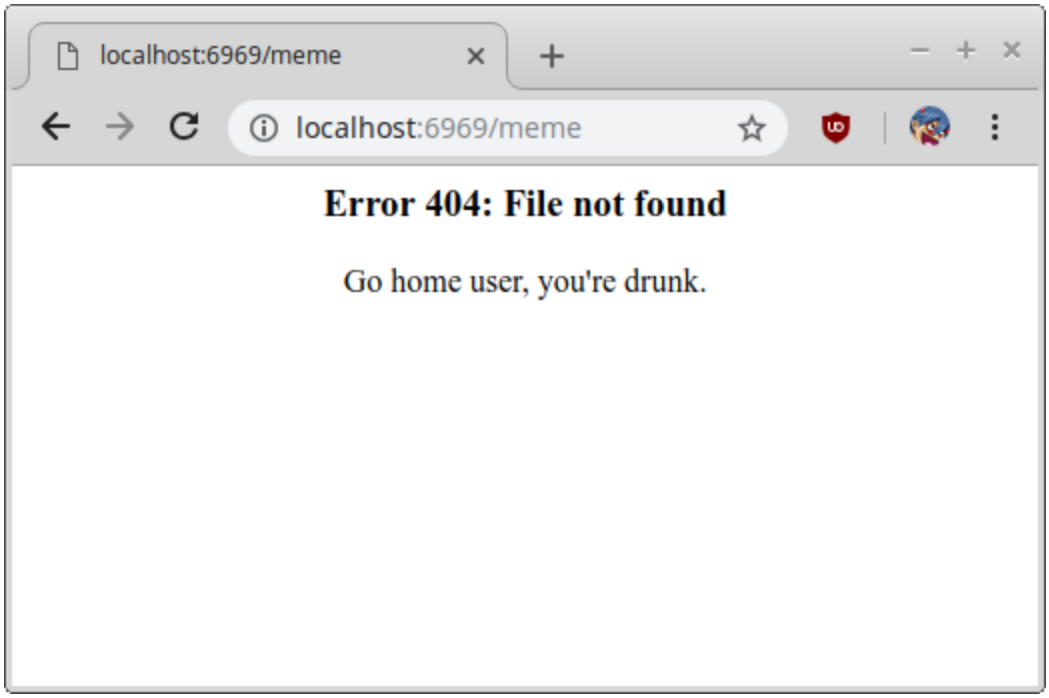
The first thing the server does upon starting up is to prepare and serve a socket for incoming client connections. Whenever the server receives a request, it prints it out on the terminal to allow the server's webmaster to analyze the data of the request. An example of a full request is:

```
GET /page1.html HTTP/1.1
Host: localhost:6969
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.
3626.81 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://localhost:6969/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

As we can see here, the requested file is /page1.html. If the file requested can be opened, a header displaying *HTTP/1.1 200 OK* is created. It is then followed up with the mime type of the requested file. Only http text files were required, but I also added jpg and png image support, as well as txt text support. The mime types of those are *text/html, image/jpg, and image/png* respectively. The requested file above is a html file. Thus, it loads the page and the following is displayed:



Should there be no specified file, the default is /index.html, which will serve as a hub to other functions of the server. This fulfills the function of html page navigation. Should an invalid file be presented, a 404 page will be displayed by default.

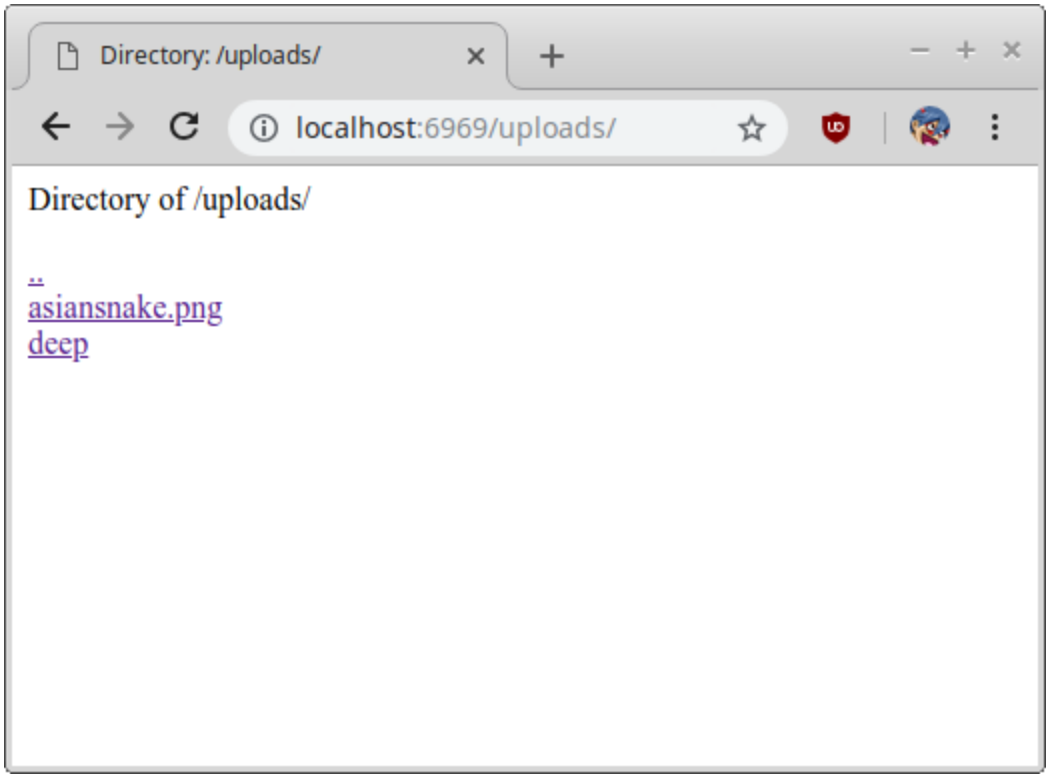


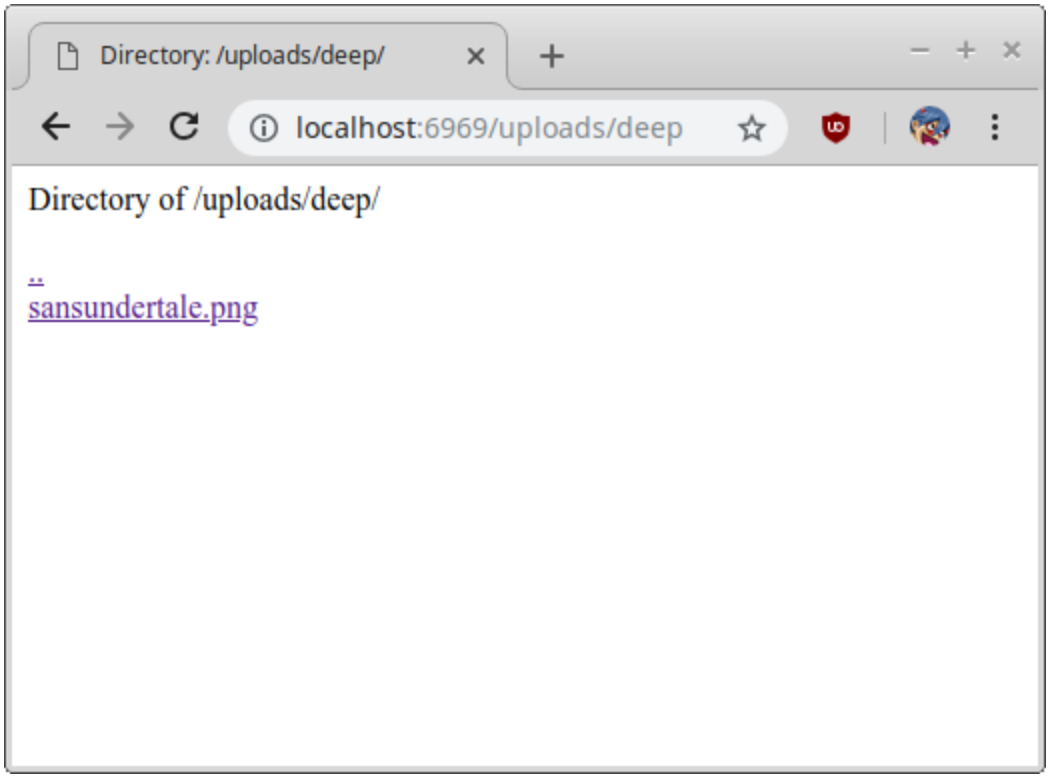
Directory Listing

The next function that was implemented was directory listing. If the server attempts to read the file, but the IsADirectoryError exception is thrown, a custom http response will be generated to go through the directory and list all the available files. The code to generate this is:

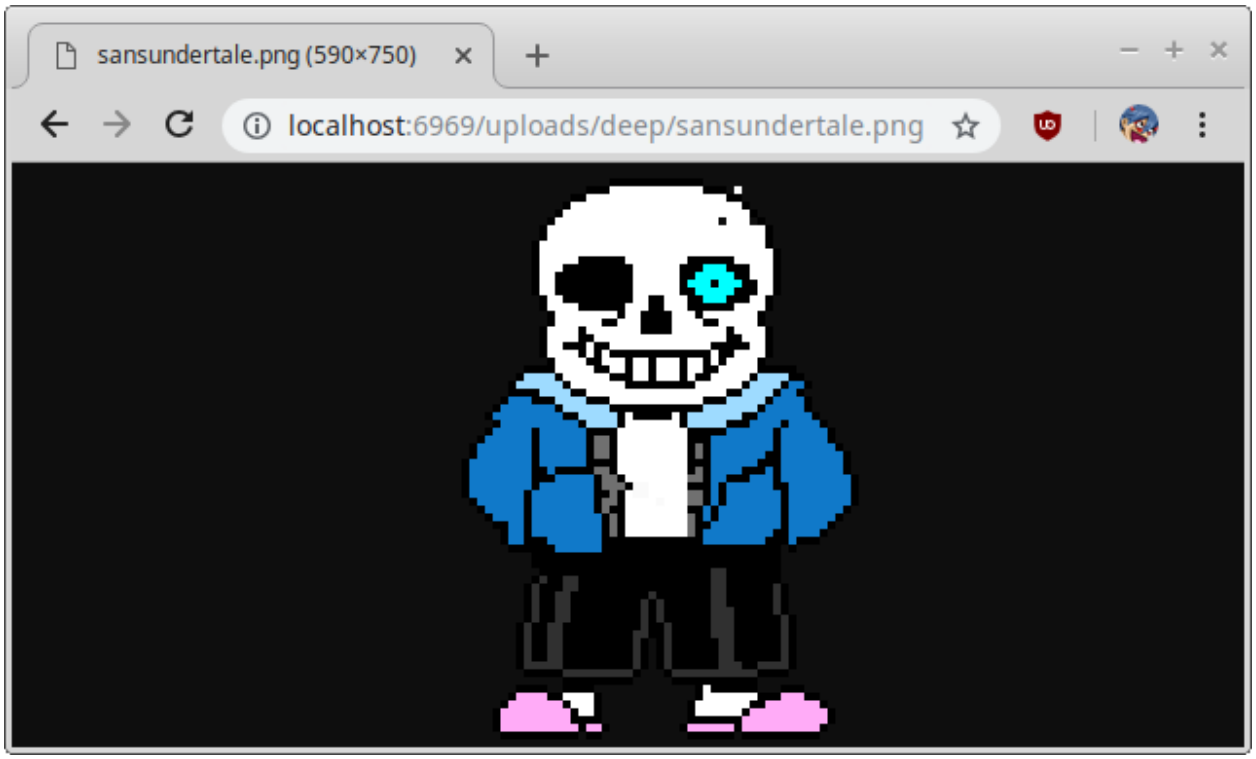
```
def create_dir_html(dir_name):
    #generates http response for a directory
    if(dir_name[-1] != '/'):
        dir_name += '/'
    cut_name = dir_name[25:]
    f = "<html>\n<head>\n<title>Directory: "+cut_name+"</title>\n</head>\n<body>\n"
    f += "Directory of "+cut_name+"<br><br>\n"
    f += "<a href = \""+cut_name+"../ \">..</a><br>\n"
    for file in os.listdir(dir_name):
        filename = os.fsdecode(file)
        f += "<a href = \""+cut_name+filename+"\">"+filename+"</a><br>\n"
    f += "</body>\n</html>"
    return f.encode('utf-8')
```

Note that the code is returned by encoding it into bytes, as that is the necessary form for a http response. In my root folder I have a folder named uploads, which contains an image *asiansnake.png* and another folder called *deep*. We can then go into *deep* and see that it contains *sansundertale.png*. These can all be navigated through, along with a .. option to go up a level.





Naturally, the files can be accessed from here as well.



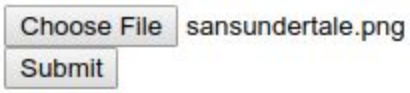
Thus, if a directory is requested to the server, its contents can be fully navigated.

File Uploads

Notice how in the previous requests, the first word was GET. To upload a file, we will have to create a POST. This is done with the following code:

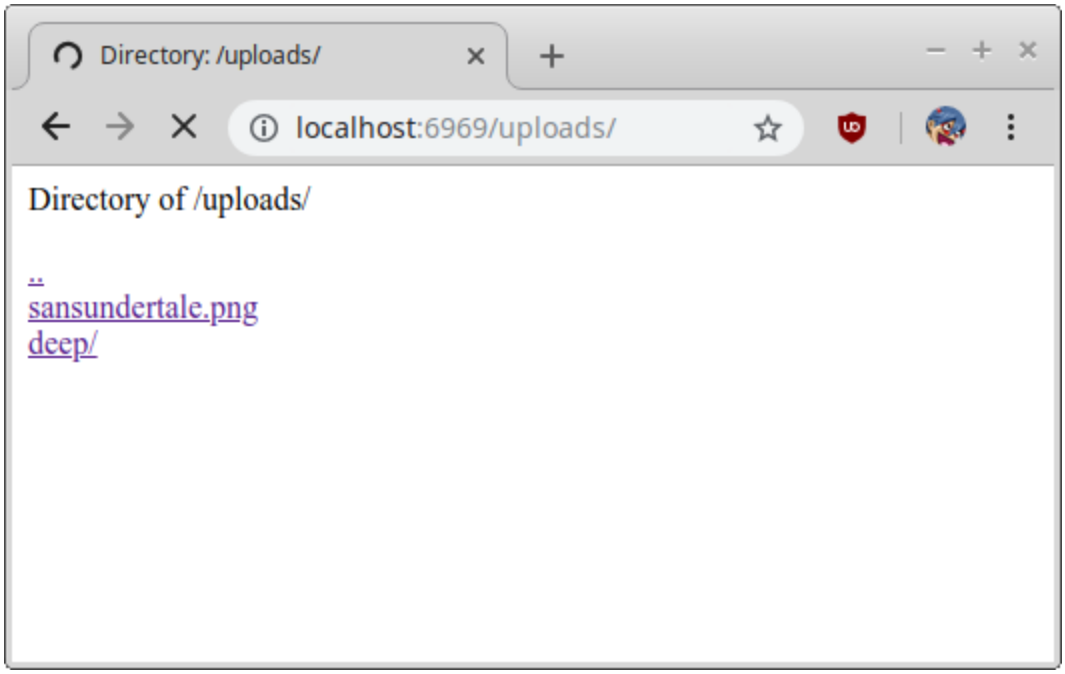
```
<html>
    <body>
        <form method="POST" action="uploads/" enctype="multipart/form-data">
            <input name="uploadFile" id="uploadFile" type="file" />
            <br>
            <input type="submit" id="submit" value="Submit">
        </form>
    </body>
</html>
```

This sends a POST request rather than a GET. The request looks like this:




```
POST /uploads/ HTTP/1.1
Host: localhost:6969
Connection: keep-alive
Content-Length: 25577
Cache-Control: max-age=0
Origin: http://localhost:6969
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryBpFTlP3hHBLhZXgV
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.
3626.81 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://localhost:6969/input.html
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

The server python code must parse whether the first word is POST or GET to determine how to handle the request in the proper way. This is done with a simple if/else so I won't add the code snippet here. It'll be available in the provided code. And because the action goes to "uploads/" then after upload it automatically loads into the uploads directory mentioned eariler. As you can see, the uploaded file *sansundertale.png* is now present in the directory (as well as the deep folder used for directory navigation testing).


Directory: /uploads/
localhost:6969/uploads/
Directory of /uploads/
..
sansundertale.png
deep/