**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

**---oOo---**



**BÁO CÁO ĐỒ ÁN THỰC HÀNH**

**LẬP TRÌNH SOCKET**

Họ và tên sinh viên 1: Ngô Đặng Gia Lâm

Mã số sinh viên 1: 19120268

Họ và tên sinh viên 2: Huỳnh Tấn Thọ

Mã số sinh viên 2: 19120383

Lớp: 19CTT2

Môn học: Mạng máy tính

Giảng viên: Chung Thùy Linh

Thành phố Hồ Chí Minh, tháng 12 năm 2020

**MỤC LỤC**

GIỚI THIỆU CHUNG 3

PHẦN 1: CÀI ĐẶT THUẬT TOÁN CHO ỨNG DỤNG WEB SERVER 4

1. Một số thành phần cơ bản 4

2. Nhận request từ client 7

2.1. Hàm getRequest(). 7

2.2. Class RequestParse 9

3. Xử lý request từ client 11

3.1. Hàm postMethod() 13

3.2. Hàm getMethod() 18

3.3. Class Response 21

3.4. Class renderfile 29

PHẦN 2: DEMO ỨNG DỤNG WEB SERVER 32

1. Khởi động và truy cập vào server 32

2. Đăng nhập vào server 34

3. Tải xuống tệp tin từ server 36

4. Một số chức năng khác 43

TỔNG KẾT 45

TÀI LIỆU THAM KHẢO 46

**GIỚI THIỆU CHUNG VỀ ĐỒ ÁN**

**1. Mục tiêu**

* Hiểu được cách hai ứng dụng giao tiếp với nhau thông qua mạng, đặc biệt là ở tầng transport – tầng vận chuyển.
* Biết thêm nhiều kiến thức về lập trình socket, kiến trúc server – client.
* Hiểu sâu hơn về các giao thức HTTP, một giao thức ở tầng application – tầng ứng dụng.

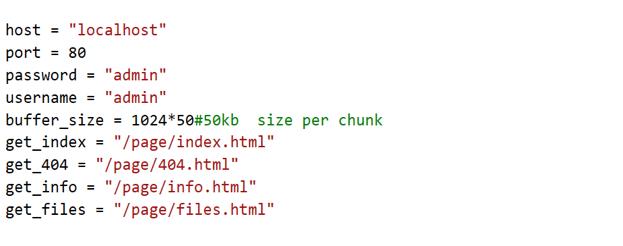
**2. Mô tả**

* Viết một ứng dụng web server sử dụng lập trình socket để được một trang web thỏa mãn các yêu cầu sau:
* Trang index.html là nơi người dùng đăng nhập.
* Từ trang index.html bên trên, người dùng sẽ nhập vào username là “admin” và password là “admin”. Sử dụng HTTP method là POST để gửi thông tin về cho server.
* Nếu thông tin đăng nhập là chính xác thì điều hướng người dùng tới trang info.html chứa hình ảnh và thông tin của các thành viên trong nhóm.
* Nếu thông tin đăng nhập không chính xác, trả về mã 404 cho trình duyệt và trả về trang 404.html để báo lỗi.
* Tạo trang files.html hiển thị danh sách các tệp tin mà người dùng được phép tải về thông qua trình duyệt. Yêu cầu Web Server trả về Transfer-Encoding là chunked thay vì Content-Length.
* Trên server, sẽ có một thư mục Download chứa các tệp tin có thể tải xuống, bên trong thư mục này sẽ không có bất kỳ thư mục con nào. Trang files.html sẽ hiển thị danh sách các tệp tin này và người dùng chọn tệp tin bất kỳ để tải xuống.

**PHẦN 1: CÀI ĐẶT THUẬT TOÁN CHO ỨNG DỤNG WEB SERVER**

**1. Một số thành phần cơ bản để khởi tạo ứng dụng**

* Trước khi đi vào các chức năng chính như đã liệt kê bên trên, trước hết ta cần thiết lập vài yếu tố cơ bản cho server.
* File index.py được xem như hàm main của chương trình, ta dùng nó để khởi tạo server cũng như thiết lập kết nối với client.
* Đầu tiên, thư viện ta dùng chủ yếu là socket. Đây là thư viện quan trọng chứa các hàm mà ta cần. Ngoài ra, ta cũng import file config.py, file này ta dùng để định nghĩa một số từ khóa quan trọng.



* + Ta import thư viện và file config.py vào file index.py.



* + Khởi tạo server với dòng lệnh

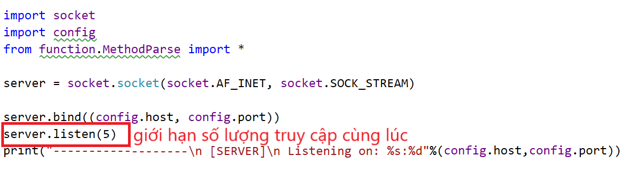
server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)



* + Kết nối server với IP và port tương ứng bằng hàm server.bind()



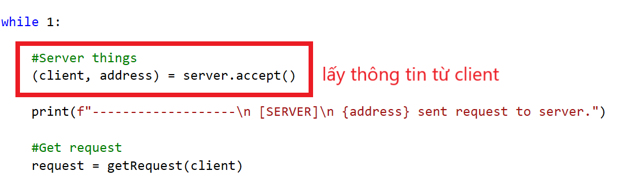
* + Ta dùng hàm server.listen() dùng để quy định số lượng máy khách tối đa có truy cập vào server cùng lúc. Ở đây, ta quy định con số này là 5.



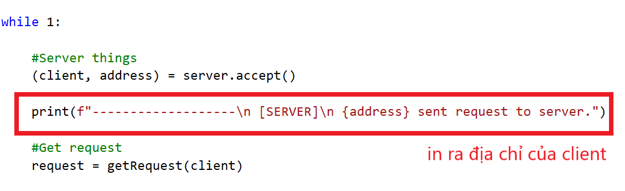
* + - Ta in thông tin về host cũng như port hiện tại của server để cho người dùng nắm bắt được rõ hơn (không bắt buộc).



* + - Dòng lệnh (client, address) = server.accept() dùng để đợi và lấy thông tin của client khi có client vào, sau đó ta có thể in địa chỉ IP của client đó.

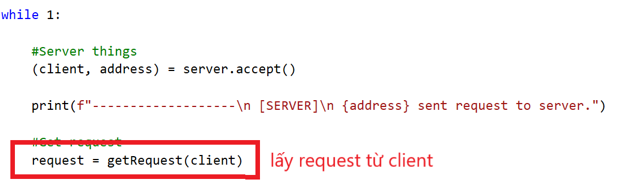


* + - In ra địa chỉ IP của client.



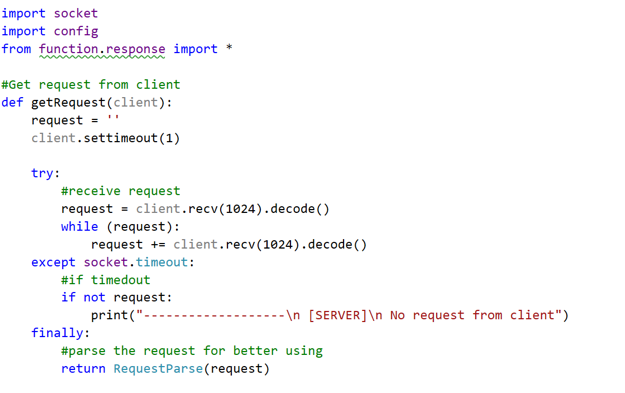
**2. Nhận request từ client**

* + - Sau khi đã đăng nhập thành công, ta sử dụng hàm getRequest() để lấy request từ client, và lưu vào biến request.



2.1. HàmgetRequest()

* + - Hàm getRequest() là một hàm ta tự định nghĩa trong file MethodParse.py với tham số đầu vào là client, có nghĩa là lấy request từ client.
    - Bên trong file MethodParse.py, ta cũng import config và thư viện là socket.



* + - Đầu tiên, ta gán biến request là rỗng và thiết lập thời gian chờ (timeout) cho client bằng hàm client.settimeout(). Ở đây, ta thiết lập là 1 giây.



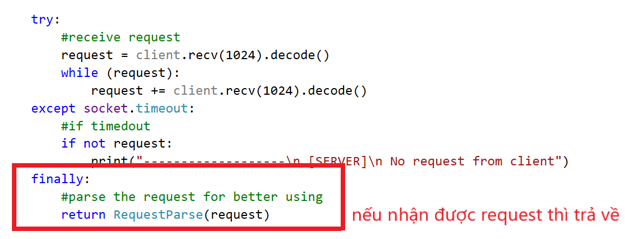
* + - Ta thử nhận request từ client xem có nhận được gì không.



* + - Nếu timeout, tức nếu hết thời gian chờ và request vẫn không có gì, ta in ra dòng thông báo và thử lại.



* + - Cuối cùng, khi đã nhận được request, ta trả về request đó dưới dạng một đối tượng thuộc class RequestParse. Class này sẽ được đề cập ngay bên dưới.



2.2. Class RequestParse

* + - Class RequestParse có một constructor với đầu vào là biến request.



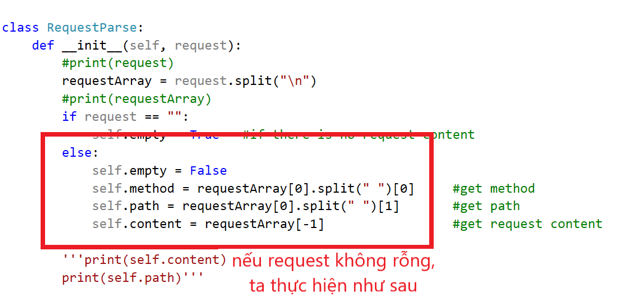
* + - Trong constructor này, trước hết, ta tách request ra thành các phần riêng biệt, với dấu phân cách là ký tự xuống dòng, và gán các thành phần được tách ra vào biến requestArray.



* + - Sau đó, ta kiểm tra điều kiện nếu request là rỗng, ta đánh dấu đối tượng của class này rỗng bằng cách gán empty = True



* Nếu request không rỗng, ta gán empty = False, đồng thời lấy method, path và content từ requestArray, chứa các thông tin ta đã tách ra từ biến request nhận vào.



**3. Xử lý request từ client**

* Quay trở lại file chính của chúng ta là file index.py, ta kiểm tra xem request sau khi ta getRequest từ client có rỗng hay không.



* Nếu request không rỗng, ta tiếp tục xét xem method của request là gì. Nếu method là POST thì thực hiện hàm postMethod() với tham số đầu vào là client và request. Method POST xảy ra khi client gửi dữ liệu cho server xử lý.



* + - * Nếu request không rỗng, và method không phải là POST, tức nó là GET, thì ta thực hiện hàm getMethod() với tham số đầu vào là client và request. Method GET xảy ra khi client yêu cầu dữ liệu từ server.



3.1. Hàm postMethod()

- Trước tiên, ta cài đặt hàm postMethod(), hàm này được định nghĩa bên trong file MethodParse.py và nhận 2 tham số đầu vào là client và request. Hàm này được dùng để xử lý request từ client nếu method của request là POST.



***a. Điều hướng đến trang files.html***

* Ta kiểm tra đường dẫn path của request, là files.html thì ta trả về trang files.html. Cụ thể như sau:



* Ta mở file có tên block.txt trên máy ở chế độ “w” và ghi vào True. File block.txt được sử dụng để xác định xem người dùng có thông qua method POST hay không.



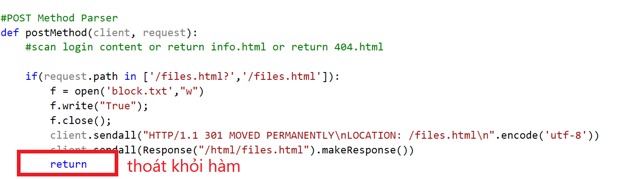
* Gửi phản hồi lại cho client, thông báo di chuyển qua vị trí /files.html với mã trả về (status code) là 301 MOVED PERMANENTLY. Đồng thời mã hóa thông báo này với định dạng UTF-8.



* Sau đó, ta tạo một đối tượng thuộc class Response từ file files.html rồi trả về đối tượng này cho client. Class Response sẽ được đề cập sau.



* Cuối cùng, ta thoát khỏi hàm bằng lệnh return.



***b. Điều hướng đến trang info.html sau khi đăng nhập thành công***

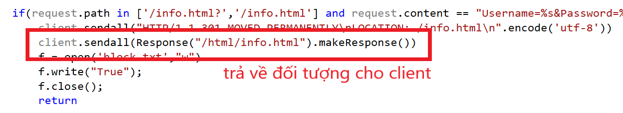
* + Nếu đường dẫn path của request là info.html và nội dung content của request trùng với username và password được định nghĩa sẵn trong file config.py, tức là đăng nhập thành công thì ta trả về trang info.html. Cụ thể như sau:



* Gửi phản hồi lại cho client, thông báo di chuyển qua vị trí /info.html với mã trả về (status code) là 301 MOVED PERMANENTLY. Đồng thời mã hóa thông báo này với định dạng UTF-8.



* Sau đó, ta tạo một đối tượng thuộc class Response từ file info.html rồi trả về đối tượng này cho client. Class Response sẽ được đề cập sau.



* Ta mở file có tên block.txt trên máy ở chế độ “w” và ghi vào True.



* Cuối cùng, ta thoát khỏi hàm bằng lệnh return.



***c. Điều hướng đến info.html khi nhấn “Trở về” từ trang files.html***

* + Về cơ bản, việc này tương tự như phần 3.1b. Nếu đường dẫn path của request là info.html và nội dung content của request là rỗng, tức là ta đang muốn quay lại trang info.html từ một trang khác thì ta trả về trang info.html. Cụ thể như sau:



* Gửi phản hồi lại cho client, thông báo di chuyển qua vị trí /info.html với mã trả về (status code) là 301 MOVED PERMANENTLY. Đồng thời mã hóa thông báo này với định dạng UTF-8.



* Sau đó, ta tạo một đối tượng thuộc class Response từ file info.html rồi trả về đối tượng này cho client. Class Response sẽ được đề cập sau.s



* Ta mở file có tên block.txt trên máy ở chế độ “w” và ghi vào True.



* Cuối cùng, ta thoát khỏi hàm bằng lệnh return.



***d. Điều hướng đến 404.html để báo lỗi***

* + Nếu tất cả những điều kiện trên đều không thỏa mãn, tức nếu đường dẫn path sai và nội dung content của request sai, ta trả về trang 404.html báo lỗi. Cụ thể như sau:
* Gửi phản hồi lại cho client, thông báo di chuyển qua vị trí /404.html với mã trả về (status code) là 301 MOVED PERMANENTLY. Đồng thời mã hóa thông báo này với định dạng UTF-8.



* Sau đó, ta tạo một đối tượng thuộc class Response từ file 404.html rồi trả về đối tượng này cho client.

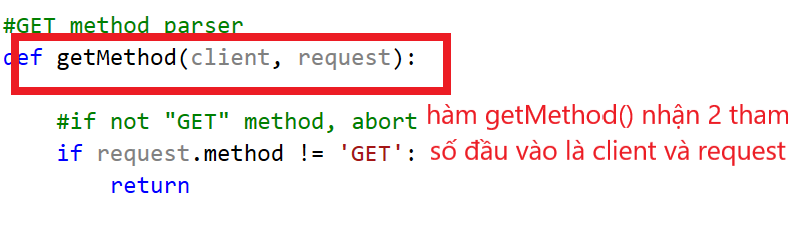


* Cuối cùng, ta thoát khỏi hàm bằng lệnh return.

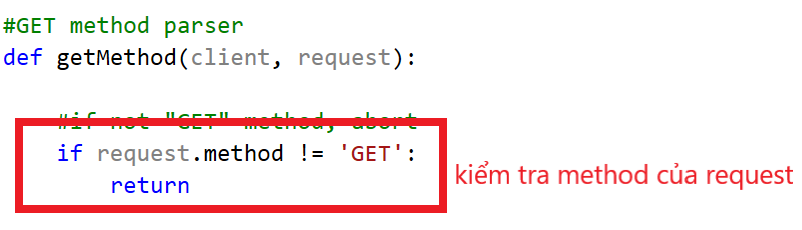


3.2. Hàm getMethod()

* + Tiếp theo, ta cài đặt hàm getMethod(), hàm này được định nghĩa bên trong file MethodParse.py và nhận 2 tham số đầu vào là client và request. Hàm này được dùng để xử lý request từ client nếu method của request là GET.



* + Đầu tiên, ta kiểm tra xem method của request hiện tại có phải là GET hay không, nếu phải thì mới tiếp tục, còn nếu không thì ta kết thúc hàm.



* + Sau khi đã kiểm tra xong, bước tiếp theo, ta tiến hành kiểm tra xem yêu cầu của người dùng của hợp lệ hay không. Nếu người dùng chưa đăng nhập nhưng cố gắng truy cập vào files.html hoặc info.html thì server sẽ chặn truy cập và trả về 404.html.
  + Cụ thể như sau:
* Ta mở file block.txt ở chế độ “r” và đọc dữ liệu từ file rồi lưu vào biến check.



* Nếu check là chuỗi ký tự “False” và nếu đường dẫn path của request là files.html hoặc info.html, thì ta chặn không cho truy cập vào trang đích bằng cách gán path của request là 404.html.





* Nếu không, ta mở file có tên block.txt trên máy ở chế độ “w” và ghi vào False.



* Kế tiếp, nếu đường dẫn path của request là “/”, tức người dùng truy cập lần đầu tiên, thì ta điều hướng người dùng đến trang đăng nhập bằng cách gán path của request là index.html.



* Cuối cùng, ta tạo một đối tượng thuộc class Response từ đường dẫn path của request rồi trả về đối tượng này cho client. Đường dẫn path này thật ra cũng là một trong các file html.



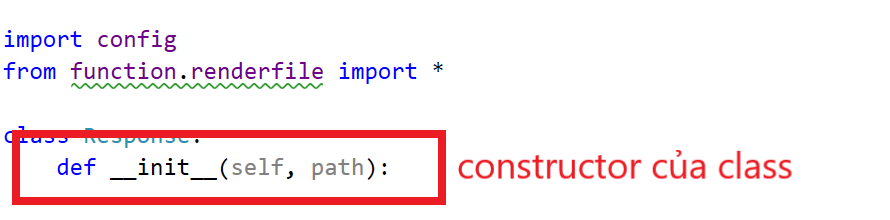
3.3. Class Response

* Như đã đề cập bên trên, các đối tượng từ class này sẽ được gửi cho client sau khi xử lý yêu cầu từ client.
* Class bao gồm 1 constructor và 1 method makeResponse(), được định nghĩa trong file response.py.

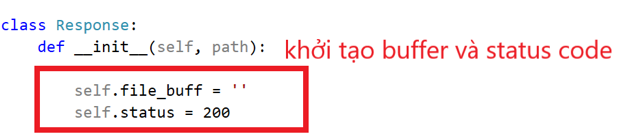


***a. Constructor***

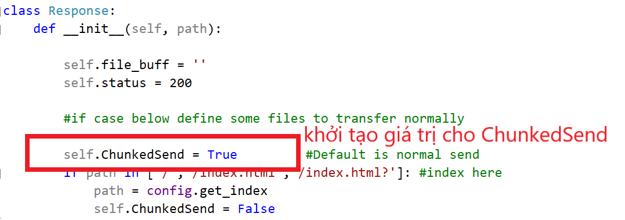
* Constructor nhận tham số đầu vào là đường dẫn path của request.



* Khởi tạo giá trị cho buffer và status code.



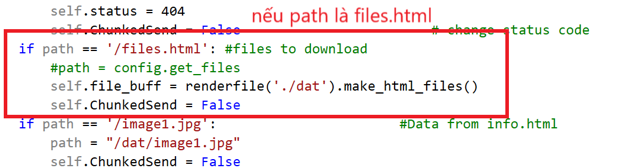
* Biến self.ChunkedSend dùng để đánh dấu rằng ta sẽ truyền dữ liệu cho client bình thường hay sử dụng chunked. Ta khởi tạo giá trị ban đầu cho self.ChunkedSend là True.



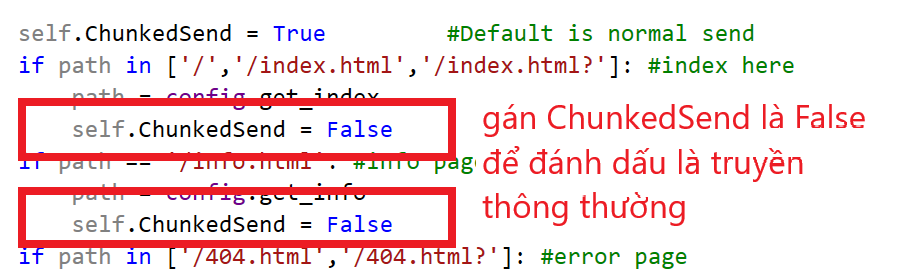
* Trong bài này, ta chỉ truyền kiểu chunked khi client thực hiện tải xuống file từ server. Do đó, nếu đường dẫn path của request là một trong số những đường dẫn sau: index.html, info.html, files.html, 404.html,…và một số nội dung khác như hình ảnh, logo,…thì không cần phải truyền kiểu chunked mà thay vào đó, ta thực hiện truyền như bình thường.



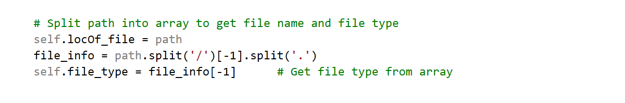
* Nếu đường dẫn path là files.html, ta gán biến self.file\_buff là một đối tượng thuộc class renderfile. Class này sẽ được đề cập sau.



* Để đánh dấu truyền kiểu bình thường, ta gán self.ChunkedSend = False.



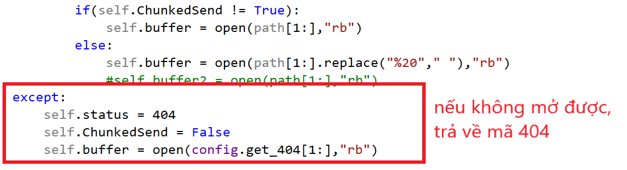
* Do path thường dẫn đến 1 file nào đó, có thể là html hoặc jpg,… Ta tách path thành mảng để lấy tên file và kiểu file.



* Sau khi có được tên file và kiểu file, ta thử mở file đó.



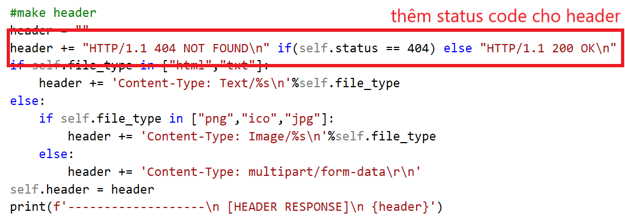
* Nếu mở không được, trả về status code là 404 đồng thời trả về file 404.html để báo lỗi.



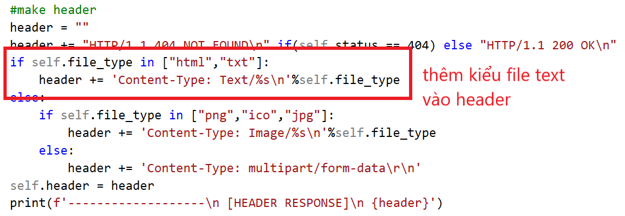
* Tiếp theo, việc ta cần làm là tạo header. Ban đầu, ta khởi tạo header là rỗng.



* Nếu status code là 404 thì ta thêm HTTP/1.1 404 NOT FOUND vào header. Nếu không, ta thêm HTTP/1.1 200 OK vào header.



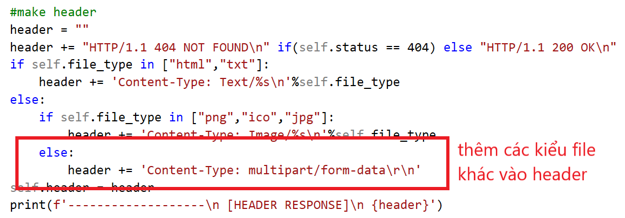
* Nếu kiểu file là html hoặc txt, ta thêm Content-Type: Text và thêm kiểu file tương ứng vào header.



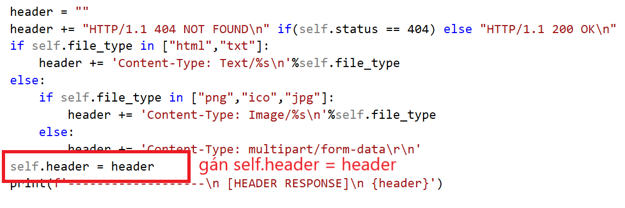
* Nếu kiểu file là jpg, png hoặc ico, ta thêm Content-Type: Image và thêm kiểu file tương ứng vào header.



* Nếu nó không là hai kiểu trên, ta thêm Content-Type: multipart/form-data vào header.

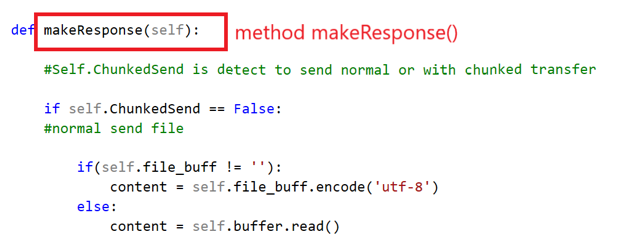


* Cuối cùng, gán self.header = header.

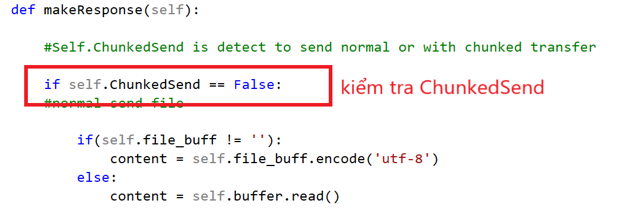


***b. Method*** *makeResponse()*

* Method makeResponse() không nhận tham số đầu vào.



* Trước tiên, ta kiểm tra xem nên truyền dữ liệu cho client bằng phương pháp nào. Ta kiểm tra biến self.ChunkedSend mà ta đã có ở constructor.



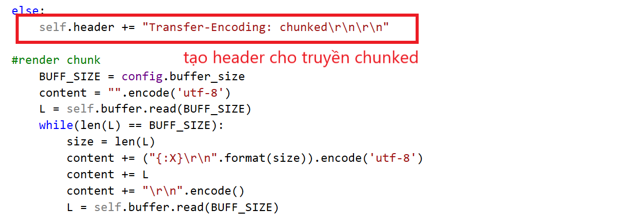
* Nếu self.ChunkedSend = false tức ta sẽ truyền theo cách thông thường.



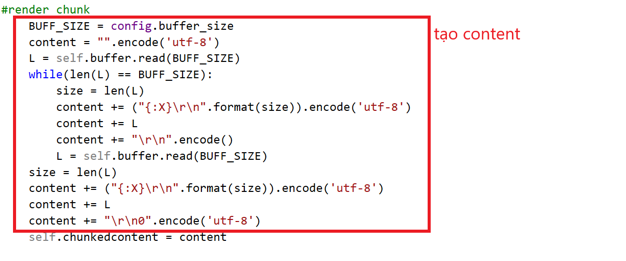
* Sau đó, ta thực hiện mã hóa header cũng như content, và trả về header.



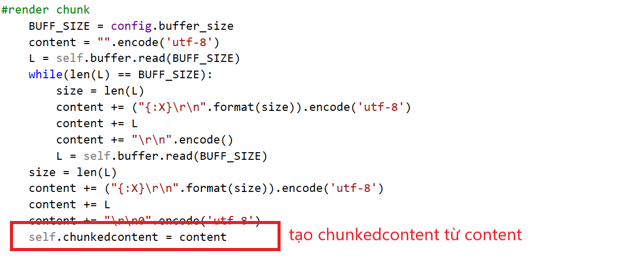
* Nếu self.ChunkedSend = true, ta sẽ truyền bằng chunked.



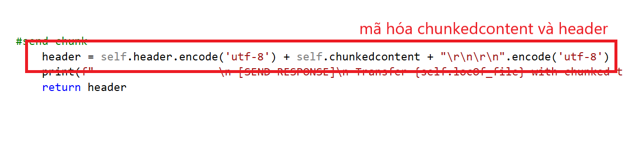
* Ta tạo content để gửi cho client.



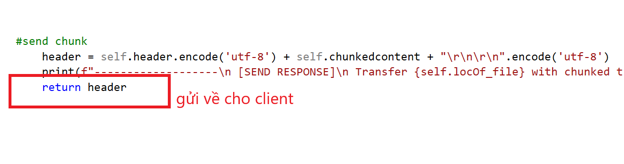
* Ta tạo self.chunkedcontent từ content.



* Mã hóa header và self.chunkedcontent.

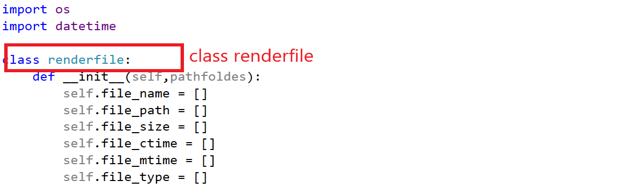


* Và cuối cùng, gửi header về cho client.



3.4. Class renderfile

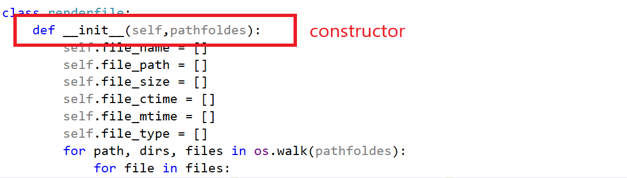
* Class renderfile được định nghĩa bên trong file renderfile.py và được dùng để hiển thị danh sách các file mà người có thể download từ server.



* Sử dụng class này, danh sách các files sẽ được cập nhật tự động khi reload trang, tiện lợi cho client trong trường hợp server thêm hoặc xóa một file nào đó.
* Class renderfile bao gồm 1 constructor để khởi tạo, và một method make\_html\_files.

***a. Constructor***

* Constructor này nhận tham số đầu vào là pathfoldes.

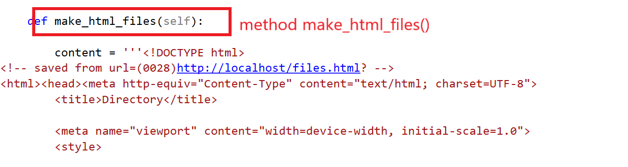


* Các bước xử lý của constructor:



***b. Method*** *make\_html\_files*

* Method này không nhận tham số đầu vào.



* Gán nội dung cho content.



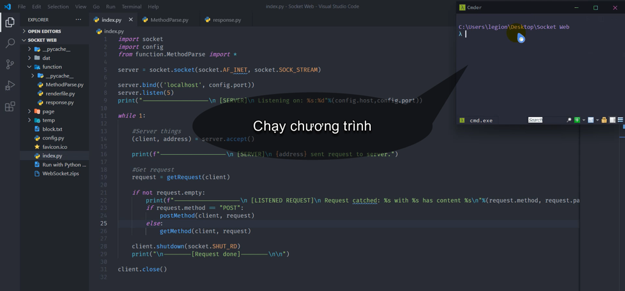
* Cuối cùng, ta trả về content.



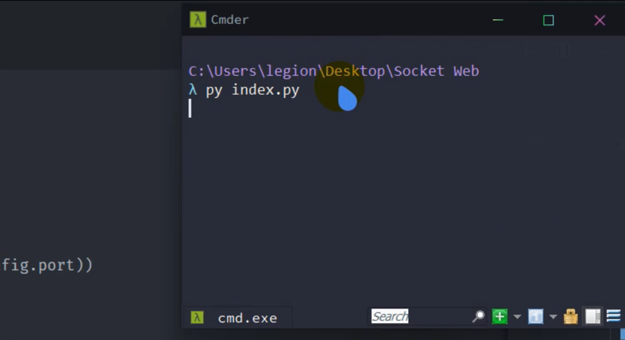
**PHẦN 2: DEMO/THỬ NGHIỆM WEB SERVER**

**1. Khởi động và truy cập vào server**

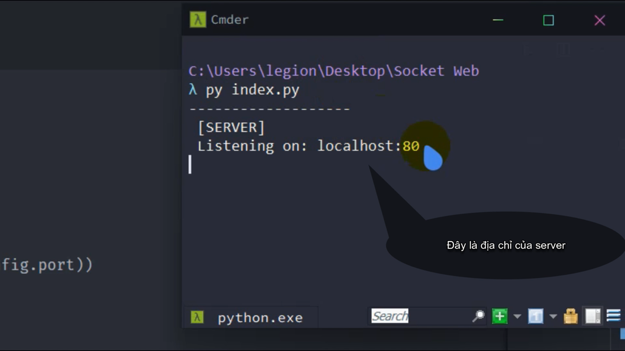
* Ta chạy Web Server này bằng Python 3.8.6 trở đi, hoặc bằng một IDE có hỗ trợ Python.



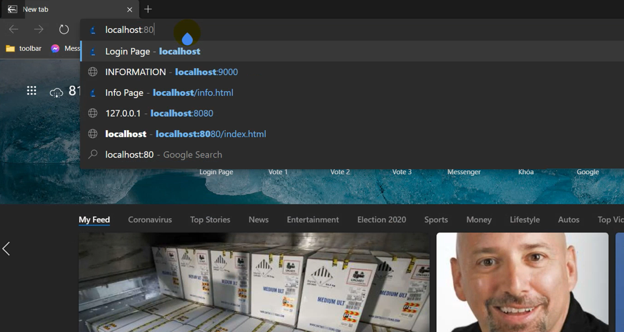
* Do index.py là hàm main, ta gõ index.py và nhấn Enter.



* Sau khi nhấn Enter, trên console hiện ra địa chỉ và port của server.

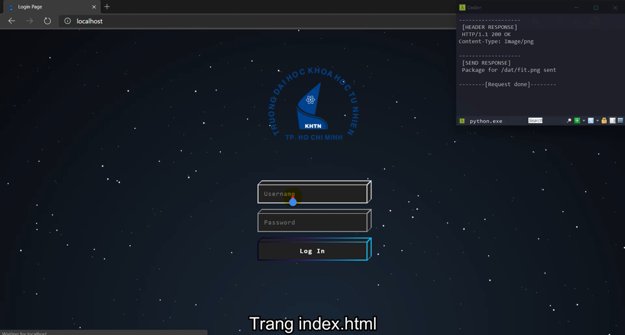


* Ta truy cập vào địa chỉ và port trên bằng trình duyệt. Ở đây địa chỉ là localhost và port là 80. Ta nhập localhost:80 vào thanh địa chỉ.

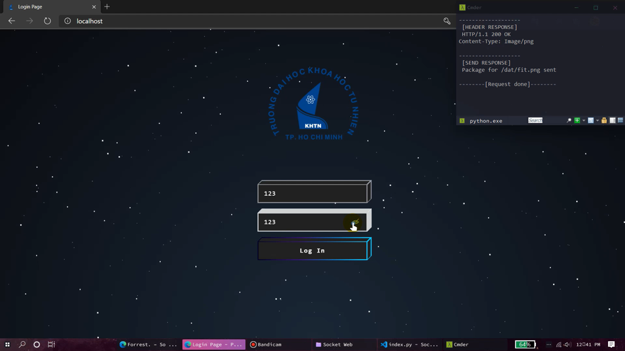


**2. Đăng nhập vào server**

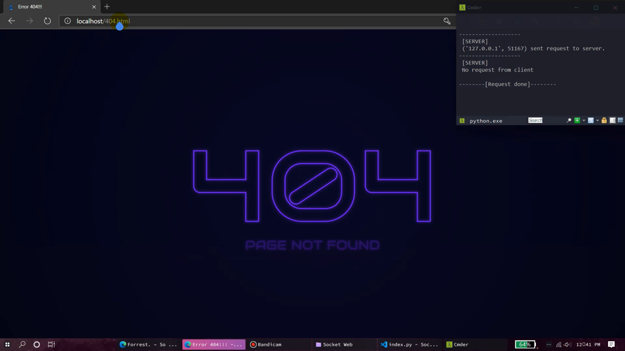
* Đây là trang index.html dùng để đăng nhập. Ta nhập username là admin và password là admin.



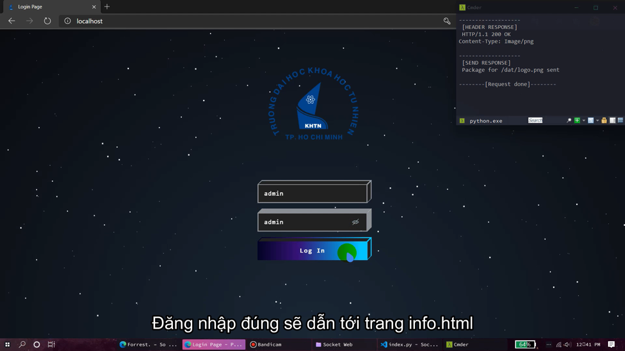
* Nhưng trước tiên, ta thử nhập sai username và password.



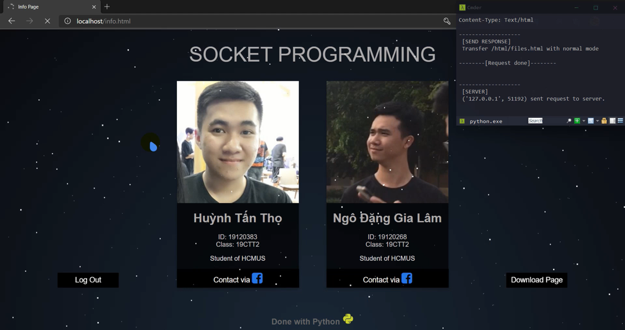
* Như ta thấy, nếu nhập sai username và password, sẽ gặp thông báo 404.



* Lần này, ta sẽ nhập đúng username là admin và password là admin.

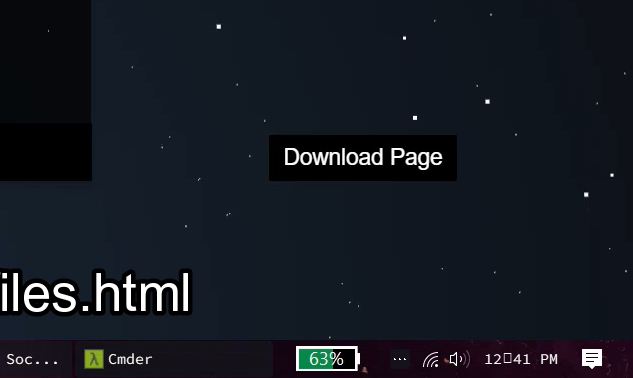


* Sau khi đăng nhập thành công, sẽ hiện ra trang info.html chứa thông tin của các thành viên trong nhóm.

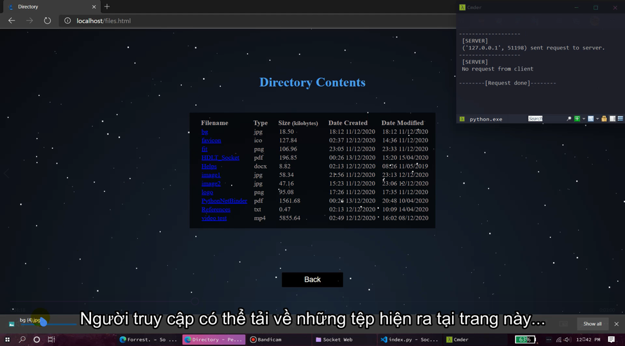


**3. Tải xuống tệp tin từ server**

* Ta nhấn nút Download Page bên góc phải để truy cập trang files.html và tải các file ta mong muốn.



* Đây là giao diện trang tải file. Nó chứa danh sách các file mà ta có thể download. Ngoài tên file, kiểu file, còn có kích chỡ file, ngày sửa đổi,…



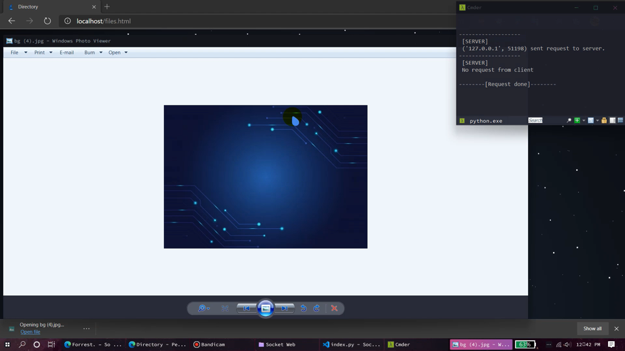
* Ở đây ta chọn file muốn download và việc download sẽ diễn ra. Ta thử tải xuống file ảnh nằm ở đầu tiên trong danh sách này.



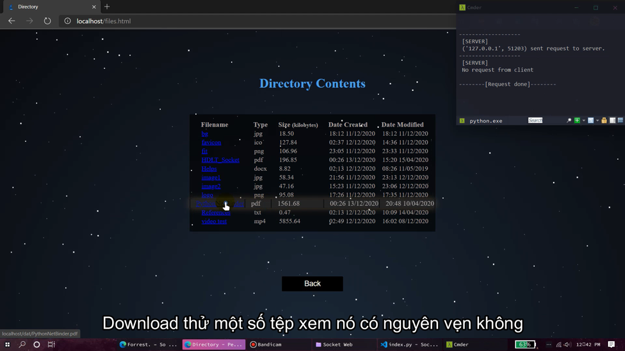
* Sau khi tải xuống thành công, ta mở file lên.



* Kiểm tra nội dung của file.



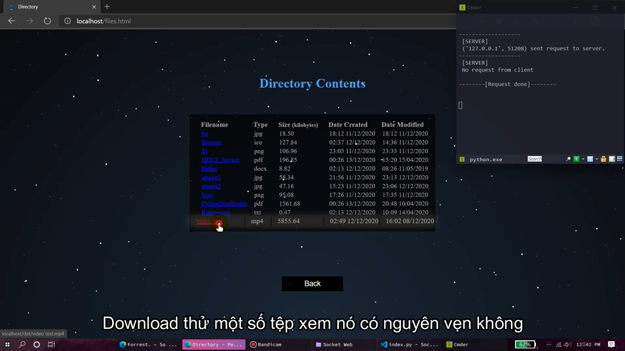
* Ta thử tải một file khác, lần này ta sẽ tải file pdf, xem có đọc được hay không. Ta chọn file pdf để download.



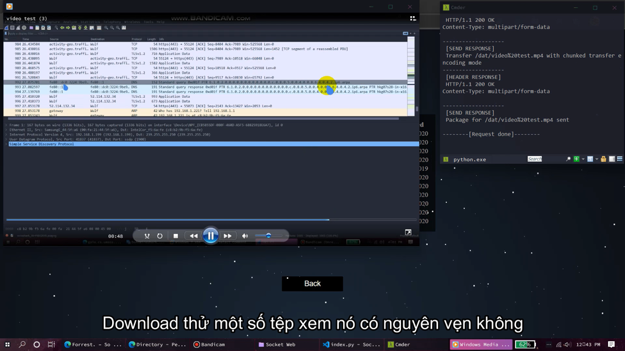
* Sau khi download hoàn tất, ta mở thử file lên xem có đọc được hay không. Như ta thấy, file pdf sau khi download vẫn đọc được bình thường.



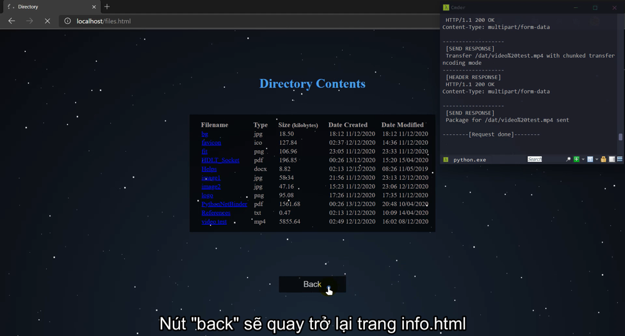
* Ta cũng thử tải xuống một file định dạng mp4 để kiểm tra.



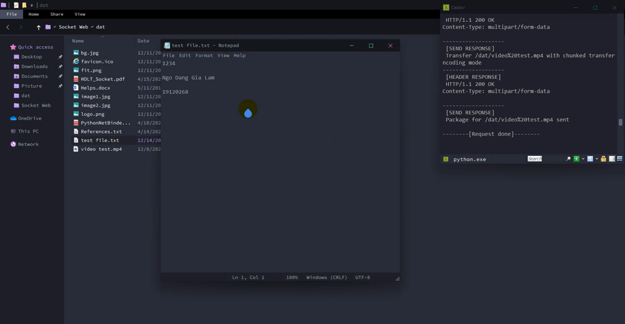
* Sau khi download hoàn tất, ta mở thử file lên xem có phát được hay không. Như ta thấy, video sau khi download vẫn phát được bình thường.



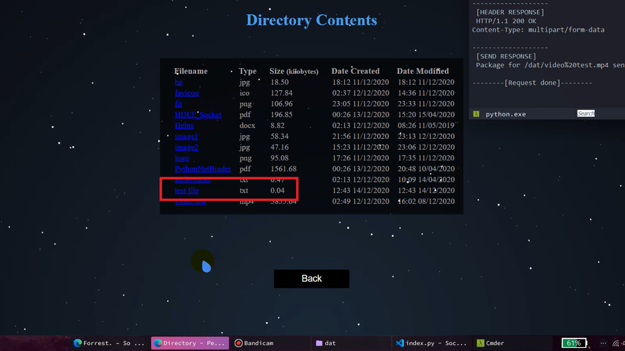
* Nếu muốn trở lại trang info.html, ta nhấn nút Back ở dưới.



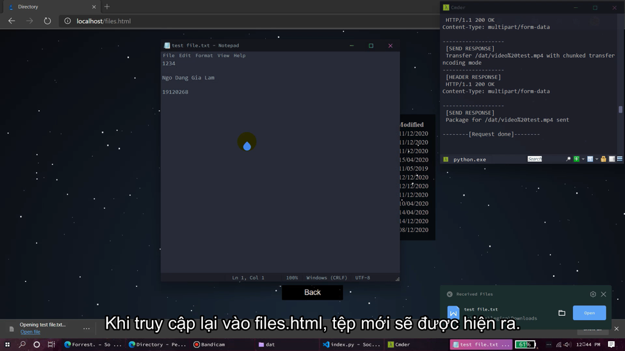
* Ngoài ra, danh sách các file có thể download là không cố định. Ta có thể thêm hoặc bớt các file cho phép người dùng tải về.
* Ta thử tạo thêm một file txt với nội dung như bên dưới và lưu lại dưới tên test file.txt.



* Sau đó, ta trở lại trang Download để kiểm tra xem file mới tạo có xuất hiện không, và như ta thấy thì nó đã được cập nhật.

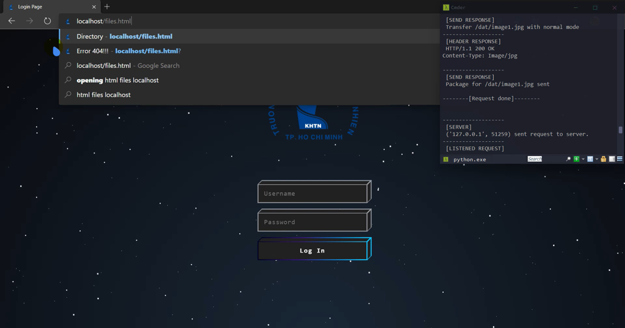


* Ta thử tải xuống file mới này, và so sánh nó với nội dung file ta đã tạo. Ta thấy nội dung là như nhau.

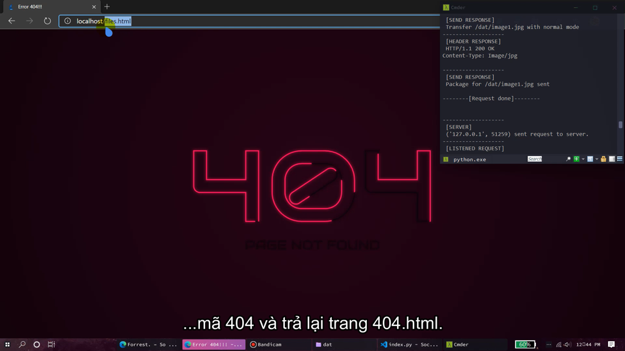


**4. Một số chức năng an toàn khác**

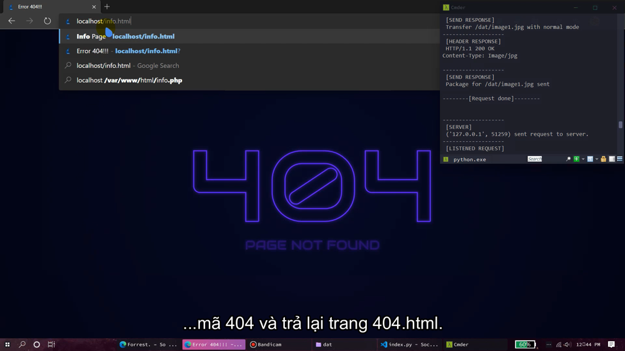
* Ngoài ra, server sẽ chỉ cho phép truy cập các trang info.html và files.html sau khi người dùng đã đăng nhập. Nếu không đăng nhập mà cố tình vào cái trang này, server sẽ báo lỗi 404. Ta thử nhập localhost/files.html vào thanh địa chỉ.



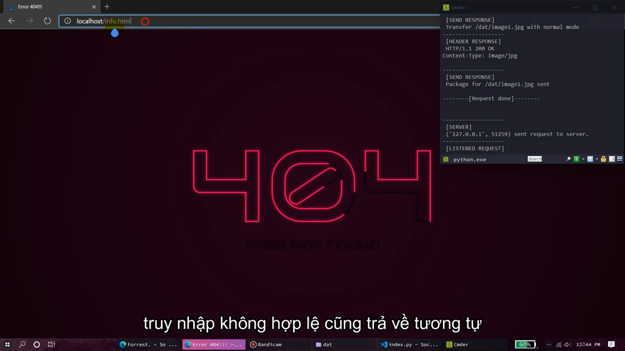
* Như ta thấy, server trả về mã 404.



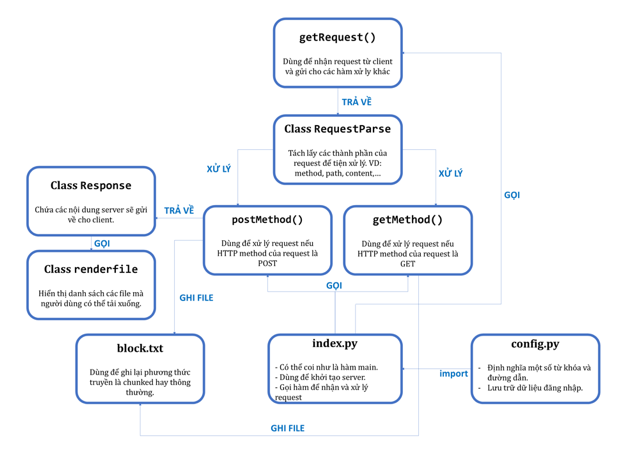
* Ta cũng thử nhập localhost/info.html vào thanh địa chỉ để kiểm tra.



* Như ta thấy, server lại trả về mã 404 do truy cập không hợp lệ.



**TỔNG KẾT LẠI CẤU TRÚC CỦA ỨNG DỤNG**



**TÀI LIỆU THAM KHẢO**

1. socket — Low-level networking interface

<https://docs.python.org/3/library/socket.html>

1. HTTP

<https://developer.mozilla.org/vi/docs/Web/HTTP>

1. Socket Programming in Python

<https://www.geeksforgeeks.org/socket-programming-python/>

1. Socket Programming in Python (Guide)

<https://realpython.com/python-sockets/>

1. How TO - Build a Website

<https://www.w3schools.com/howto/howto_website.asp>

1. HTML Tutorial

<https://www.w3schools.com/html/default.asp>

1. Redirections in HTTP

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Redirections>

1. Transfer-Encoding

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Transfer-Encoding>

1. Chunked transfer encoding

<https://en.wikipedia.org/wiki/Chunked_transfer_encoding>

1. Content-Type

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type>

1. Python Socket Programming Tutorial

<https://www.youtube.com/watch?v=3QiPPX-KeSc>

1. Python program to send and receive file using sockets

<https://www.youtube.com/watch?v=UOlxV47NUJc>

1. How To Create a Simple Web Server Using Python and the http.server Module

<https://www.youtube.com/watch?v=kogOfxg1c_g>

1. Python Requests Tutorial: Request Web Pages, Download Images, POST Data, Read JSON, and More

<https://www.youtube.com/watch?v=tb8gHvYlCFs>

1. Send simple HTTP request with HTML submit button

<https://stackoverflow.com/questions/45697176/send-simple-http-request-with-html-submit-button>

1. Building a Simple HTTP Server in Python

<https://www.youtube.com/watch?v=PNt8zXl7EJ0>

1. Send file to a browser using python socket programming

<https://stackoverflow.com/questions/40144535/send-file-to-a-browser-using-python-socket-programming>

1. Can a html button perform a POST request?

<https://stackoverflow.com/questions/16036041/can-a-html-button-perform-a-post-request>

1. Why is host aborting connection?

<https://stackoverflow.com/questions/1472876/why-is-host-aborting-connection>