# 236369

# Winter 2021-2022

# Managing Data on the World-Wide Web

## Assignment 1 – Basic Web Server in Python

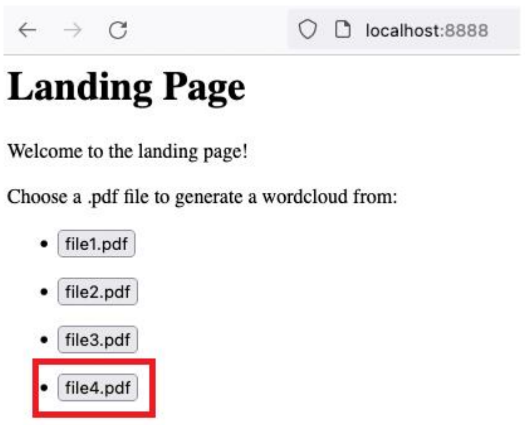
In this assignment, you will implement a simple HTTP server that generates word clouds for pdf files.

Assignment is 10% Takef.

Submission date: 14.11.2021 23:55 PM

## Introduction

The server you will implement in this assignment is a server that displays the pdf files located in a specific directory on the server and interactively generates word clouds from these files.

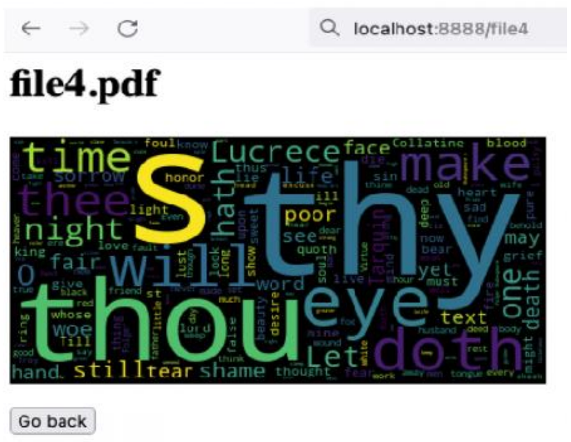


Landing Page

Welcome to the landing page!

Choose a .pdf file to generate a wordcloud from:

- file1.pdf
- file2.pdf
- file3.pdf
- file4.pdf



We will implement this server using **basic python communication – sockets.**

## Implementation Requirements

The server's implementation should use only the sockets module for networking and only TCP sockets. The server should run on the "localhost" IP and port no. 8888.

The server will work as follows: receives a GET request, whose URL represents a pdf file name on the server, for example, localhost: 8888/PdfFile, and the server will return an HTML page with an image that is generated online by the server in PNG format.

Remember to always close sockets after the connection ends and when sending an error.

### HTTP Requests

All the requests and responses correspond to the HTTP/1.1 protocol. Assume that the requests' protocol is always HTTP/1.1 and that the request has a valid structure.

The connection should be non-persistent.

The request line of the HTTP request has the following form:

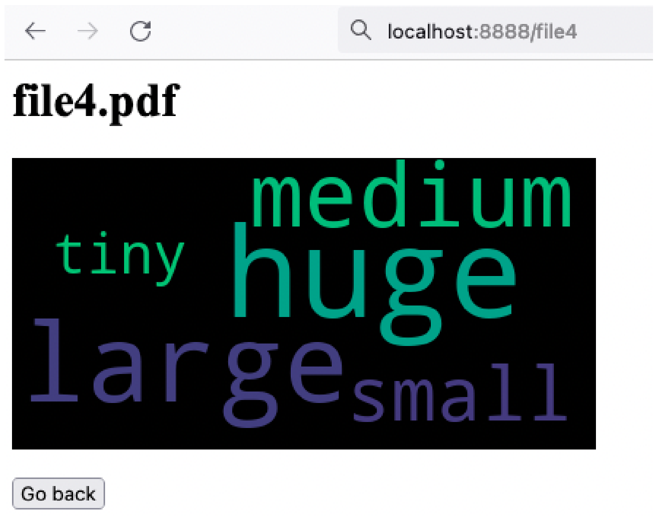
GET URL(without hostname) HTTP/1.1\r\n

The incoming requests will not always be GET requests and the server should deal with these scenarios.

### Page Format

The main page should be a valid HTML document that contains links to the different PDF files that are located under the folder 'pdfs' (including inner folders within, and so on). The folder 'pdfs' is located under the current directory, where the server code is running.  Pdf files are files with the suffix ".pdf" in their names.

The file name can be organized in tables\list\etc for displaying the data, but it must have the option to navigate from the main page to the pages with correspondent word clouds of specific pdf. The word cloud pages should contain a "previous" link to



the main page, text that states the file's name, and the word cloud. For example:

The design is for your choice as long as it functionally supports all the requirements:

1. The main page contains the names of the pdf files located under the directory "pdfs" under the current directory and allows navigation to pages displaying word clouds corresponding to those files.
2. Pages with the URL "localhost: 8888/<filename >" that display:
   a. A word cloud generated by the server for the file with the name "< filename >.pdf" corresponding to the URL.
   b. Text that states the file's name.
   c. A link to the main page.

Keep in mind that it is still allowed to send Legal HTTP/1.1 requests to the server outside of the browser!

## Generating Word Clouds

Word clouds are images composed of words used in a particular text, in which the size of each word indicates its frequency or importance.

You are provided with a utility function ***generate_wordcloud_to_file*** in ***hw1_utils.py*** for generating word clouds. Our server should create word clouds using that function and the text in the correspondent pdf file after filtering out stop words using the stop words list available in the provided file – ***stopwords.txt*** .

## Error Management

You should report errors to the client using the HTTP status codes as following:

1. A valid, error-less response should have the status 200.
2. If the requested file name does not exist **or** it is not a pdf file **or** there is no such file as "pdfs" under the current directory, the status should be 404.
3. If the request type is different from GET, the status should be 501.
4. In any other error, the status should be 500.

## Tips:

1. Start with implementing a basic server that just returns the home page as a string, and then improve it.
2. You are provided with a utility file that can parse and build simple HTTP requests\responses
3. **You should use python packages for extracting information from PDF documents. For example: "pdfminer.six"**
4. You can use *urllib* to help you parse and return URLs.
5. You can easily test with your browser or some *curl* command.

## Provided Files

***stopwords.txt*** - list of stop words to filter out from a pdf file's text before generating a word cloud. (Origin: https://www.ranks.nl/stopwords)

***hw1_utils.py*** - 2 utility functions, one for for generating word clouds and the second is for decoding http request as python dictionary.

***hw1.py*** – Here goes your implementation of the server. This file is supposed to be filled and submitted.

## Submission

You should submit a **zip** named *"hw1_<id1>_<id2>.zip"* (for example: *"hw1_123456789_987654321.zip"*) containing all the files needed for the server to work. The file: ***hw1.py*** should be the main script which runs the server.

We will run your implementation of the server with the follow command line:

- ➢ python hw1.py

- Do not submit the files ***stopwords.txt*** and ***hw1_utils.py***

**Good Luck!**