ENGR 212 Programming Practice
Mini Project 6
December 20, 2016

In this mini project you are going to develop a Python program that will help you search the contents of text files in your computer, and list the files satisfying the search criteria specified by the user. Details regarding the requirements are as follows:

- Your program will have a graphical user interface (GUI) which will look like as shown in Figure 1.

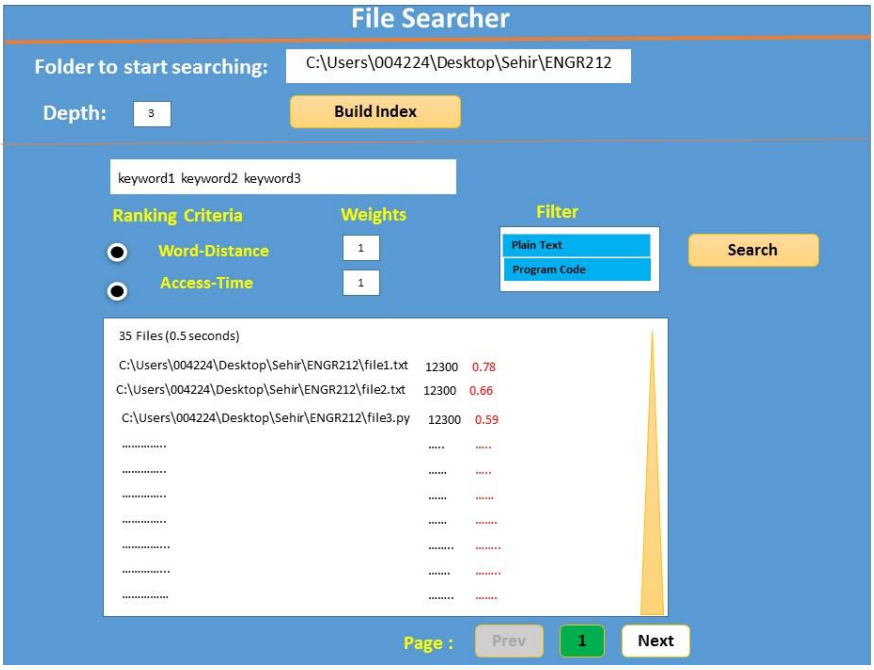

Figure 1

The user shall provide the path of an existent folder on the computer which will be used as the starting point for searching. Your program should index all text files in that folder and all subfolders under it, within a user-specified depth, when the user presses the "Build Index" button. If no depth value is specified, your program should assume an appropriate default value.

- While processing each folder, the program shall extract the time of the most recent access to text files being processed, and the type of the file (program code in some programming language, or plain text) in addition to indexing the contents of the file; and store them all in a database. You need to design a database using shelve module. You will decide what dictionaries you will maintain, and what their structures would be. You may want to see ***mysearchengine.py*** as an example (e.g., indexing related functions, etc.), but you would have to come up with a modified design to solve the current problem. Note that you are indexing individual files in a directory hierarchy rather than web pages interlinked with hypertext.
- Whenever the index building step is completed, the user can start searching for text files by providing several keywords. In the search box provided, the user will type one or more search

keywords. Whenever the search button is pressed, the program should find all files containing all of the search terms typed, among the previously indexed files, and print their relevant information within the textbox.

- The program should offer two different ranking measures to be used for determining the order of the files in the result area.
    - Word distance based measure: You need to write a method that will compute the word distance-based score. More specifically, let's say that a user has searched with three keywords, "word1 word2 word3". Assume that the file contains all three words; and there are 10 words between word1 and word2, and 7 words between word2 and word3. Then, the word-distance-based score of that file is 10 + 7 = 17. If there are multiple occurrences of any of the keywords in a file, then the computed word-distance score of the file should be the minimum value among the several possible values calculated. You need not consider the relative ordering of the keywords within a file, i.e., word2 appearing before or after word1 need not make any difference with regard to the calculation of the word distance score of the file.
    - Access time measure: You shall use the most recent access time of the file as another ranking measure. Again, you need to normalize the scores as stated above (you determine whether small is better or higher is better).
    - In order to combine the above two scores, you need to take into account the weights that the user will provide (the default weight should be 1 for each measure). You may want to see mysearchengine.py for a similar method to adapt for your needs.

- ***File type filter***. The user should be able to search in either of the two categories supported by the program. In a listbox, you should provide the user with two types (Program Code, Plain Text) for him/her to select from. This listbox should allow multiple selection, and both types should be selected by default.

- If the user does not do one or more of the following, your program should generate a warning message with a proper text:
    - give a starting folder which does not exists.
    - provide at least one keyword,
    - choose at least one ranking measure,
    - provide the weights if multiple ranking measure is selected,
    - choose at least one file category.

- Next, the user will click on "Search" button. Your program should search your database for files that contain all the provided keywords filtered and ranked according to the above settings.
    - At the top, you should let the user know how many papers matched his/her query, and how long the search took.
    - Search results should be paginated with 10 results on each page (except the last one which may have less). A scrollbar should be used if 10 search results do not fit into the visible part of the search result area. You should have pagination buttons at the bottom (e.g., Previous and Next) as shown in Figure 1 to navigate between different pages. Between these buttons, there should be a label showing the current page number (which is 1 by default).

- Search results should be numbered starting from 1. For each file in the search result, you should provide its file size in bytes, as well. The computed ranking score should be also listed next to the file as shown in Figure 1.

Can you provide any further pointers that may be helpful?:

- You **should not** import ***mysearchengine.py***. You should transfer the parts that you would need or modify into your own code file.
- To determine the type of a file, install, import, and use the module named ***Python Magic***. Do not consider the extension of a file to determine its file type. Python Magic is capable of understanding the specific type of a file by looking into its headers, etc. Following link contains instructions on installing, and using Python Magic. Please follow the installation instructions specific to your own computer's operating system (Windows, Mac-OS, or Linux) (see https://github.com/ahupp/python-magic). In addition to determining the type of the given file (text, image, etc.) Python Magic has the ability to understand whether a given text file is a program written in a programming language, or plain text. If you have problem installing and using the Python Magic module, ask for help from the assistants or the instructor.
- Use the ***os*** module of Python for file and directory related operations, such as changing directories, listing files in a directory, getting the statistics (size, access time; you can use ***os.stat()***) about a file, etc.
- You may use Text widget for the search result area. You may want to use ***tag_add***, ***tag_modify***, etc. methods of Text widget for color and font configuration of particular regions on the search results (see http://stackoverflow.com/questions/14786507/how-to-change-the-color-of-certain-words-in-the-tkinter-text-widget).
- To measure the elapsed time, you may see the following: http://stackoverflow.com/questions/3620943/measuring-elapsed-time-with-the-time-module

Warnings:

- **Do not** talk to your classmates on project topics when you are implementing your projects. **Do not** show or email your code to others. If you need help, talk to your TAs or myself, not to your classmates. If somebody asks you for help, explain them the lecture slides, but do not explain any project related topic or solution. Any similarity in your source codes will have **serious** consequences for both parties.
- Carefully read the project document, and pay special attention to sentences that involve "**should**", "**should not**", "**do not**", and other underlined/bold font statements.
- If you use code from a resource (web site, book, etc.), make sure that you reference those resource at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so **may result in** plagiarism investigation.
- Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is **important** to understand all the details in your submitted code. You may be interviewed about any part of your code.

How and when do I submit my project? :

- Projects may be done individually or as a small group of two students (doing it individually

is recommended). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for details).

- Submit your own code in a **single** Python file. Name it with your and your partner's first and last names (see below for naming).
  - If your team members are Deniz Barış and Ahmet Çalışkan, then name your code file as deniz_baris_ahmet_caliskan.py (Do **not** use any Turkish characters in file name).
  - If you are doing the project alone, then name it with your name and last name similar to the above naming scheme.
- Submit it online on LMS (Go to the Assignments Tab) by **17:00 on December 30, 2016**.

  Late Submission Policy:

  - -10%: Submissions between 17:01 – 18:00 on the due date
  - -20%: Submissions between 18:01 – midnight (00:00) on the due date
  - -30%: Submissions which are 24 hour late.
  - -50%: Submissions which are 48 hours late.
  - Submission more than 48 hours late will not be accepted.

Grading Criteria? :

Code Organization			Functionality						
Meaningful variable names (%5)	Classes and objects used (%5)	Sufficient commenting (%5)	Compiles % Runs? (10)	GUI Design (10)	Reading files, building the index (20)	Filtering out files based on category (15)	Searching with single or multiple keywords and showing results properly in pages (20)	Ranking properly (20)	Generation of appropriate warning messages (20)

- Interview evaluation (your grade from interview will be between 0 and 1, and it will be used as a coefficient to compute your final grade. For instance, if your initial grade was 80 before the interview, and your interview grade is 0.5, then your final grade will be 80*0.5 = 40). Not showing up for the interview appointment will **result in** grade 0.

Have further questions? :

- Contact your TA's during their office hours, and please do NOT hesitate to ask the instructor also (by e-mail or in-person), about anything related to the project.

Good Luck!