

**Visvesvaraya Technological University
Belagavi-590018, Karnataka**



A Mini Project Report on

“Document Processing using keywords”

Submitted in partial fulfilment of the requirement for the
File Structures Laboratory with mini project [17ISL68]

**Bachelor of Engineering
in
Information Science and Engineering**

Submitted by
SNEHA BR [1JT17IS041]
Under the guidance of Mr.Vadiraja A
Asst.Professor,Department of ISE



**Department of Information Science and Engineering
Jyothy Institute of Technology
Tataguni, Bengaluru-560082**

Jyothy Institute of Technology
Tataguni, Bengaluru-560082
Department of Information Science and Engineering



CERTIFICATE

Certified that the mini project work entitled “**Document processing using keywords** ” carried out by **Sneha BR[1JT17IS041]** bonafide student of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering in Information Science and Engineering** department of the **Vishvesvaraya Technological University, Belagavi** during the year **2019-2020**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the said Degree.

Prof. Vadiraja A
Guide, Asst. Professor
Dept. Of ISE

Dr. Harshvardhan Tiwari
Associate Professor and HoD
Dept. Of ISE

External Viva Examiner
Signature with Date :

- 1.
- 2.

ACKNOWLEDGEMENT

Firstly, I am very grateful to this esteemed institution “**Jyothy Institute of Technology**” for providing me an opportunity to complete my project.

I express my sincere thanks to our Principal **Dr. Gopalakrishna K** for providing me with adequate facilities to undertake this project.

I would like to thank **Dr. Harshwardhan Tiwari, Professor and Head of Information Science and Engineering** Department for providing his valuable support.

I would like to thank my guide **Mr. Vadiraja A, Asst. Prof.** for his keen interest and guidance in preparing this work.

Finally, I would thank all my friends who have helped me directly or indirectly in this project.

Sneha BR [1JT17IS041]

ABSTRACT

A File Structures is a combination of representation of data in files and operations for accessing the data. It allows applications to read, write, annotate and modify data. A File Structure is a combination of representations for data in files and of operations for accessing the data. The project titled “**Document processing using keywords**”, where it helps to search keywords in a text file. In this process a large text file is exposed to perform operations like search, modify, annotate and show primary indexing where search helps to find keywords in a file, modify helps altering the information, the user can get the meaning of any word which is done under annotation and primary index builds the index of the words present in the text file. This project shows the basic operations performed on the large text file. The purpose of this project is to reduce the time consumed to perform these operations by the help of computerized equipments.

TABLE OF CONTENTS

SL NO	Description	Page no
1	Chapter:1 Introduction	
1.1	Introduction to File Structures	1
1.2	File System	1
1.3	Introduction to Python	1-2
1.4	Document Processing and Indexing	2
2	Chapter:2 Requirement Analysis and Design	
2.1	Domain Understanding	3
2.2	Classification of requirements	3
2.3	System Analysis	3
2.4	Block Diagram	4
3	Chapter:3 Implementation	
3.1	Indexing	5
3.2	Searching	6
3.3	Modification	7
3.4	Annotation	8-9
4	Chapter:4 Results and snapshots	
4.1	Results of Indexing	10
4.2	Results of Searching	11
4.3	Results of Modification	12
4.4	Results of Annotation	13
4.5	Graph(Time Analysis)	14-15
5	Conclusion	16

CHAPTER 1

INTRODUCTION

1.1 Introduction to File Structures

A File is a collection of data stored on mass storage. File Structure will specify the logical structure of the data ,that is the relationships that will exist between data items independently of the way in which these relationships may actually be realized within any computer.For example, if we were to store a tree structure on a magnetic disk, the physical organization would be concerned with the best way of packing the nodes of the tree on the disk given the access characteristics of the disk. File Structures is the Organization of Data in secondary storage Device in such a way that minimize the access time and the storage space.Secondary storage such as disks can pack thousands of megabytes in a small physical location.Since the details of the representation of the data and the implementation of the operations determine the efficiency of the file structure for particular applications,improving these details can help improve secondary storage access time.Data Processing from a computer science perspective can have storage of data,organization of data,Access to data. This will be built on the knowledge of data structures.both data structures and file structures involve representation of data with operations for accessing data.the difference is data structures deal with data in main memory whereas file structures deal with data in secondary storage device(file).

1.2 File Systems

A file system is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk. The word is also used to refer to a partition or disk that is used to store the files or the type of the file system.In computing, a file system controls how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with no way to tell where one piece of information stops and the next begins. Consider an UNIX file system. Most UNIX file system types have a similar general structure, although the exact details vary quite a bit. The central concepts are super block,inode, data block, directory block and indirection block. The super block contains information about the file system as a whole, such as its size (the exact information here depends on the file system). An inode contains all information about a file, except its name. The name is stored in the directory, together with the number of the inode. A directory entry consists of a filename and the number of the inode which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file. Linux supports several types of file systems.

1.3 Introduction to Python

Python is a dynamic, interpreted (bytecode-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs. Like C++ and Java, Python is case sensitive so "a" and "A" are

different variables. The end of a line marks the end of a statement, so unlike C++ and Java, Python does not require a semicolon at the end of each statement. Comments begin with a '#' and extend to the end of the line. Python source files use the ".py" extension and are called "modules." One unusual Python feature is that the whitespace indentation of a piece of code affects its meaning. A logical block of statements such as the ones that make up a function should all have the same indentation, set in from the indentation of their parent function or "if" or whatever. If one of the lines in a group has a different indentation, it is flagged as a syntax error.

1.4 Document Processing and Indexing

The definition of a document is quite restrictive for it specifies a substance (paper, in our case an electronic document), the method of making marks on it (writing or printing), and its use (furnishing information or evidence, legal or official). Document processing is a very minor aspect in the File Structure field. Using concepts like indexing and hashing we can expand the uses and application of File Structures. Index or indexed file is structure containing a set of entries, each consisting of a key field and a reference field, which is used to locate records in a data file. The part of an index which contains keys is called the key field and the part of an index which contains information to locate records is called reference field. By indexing we impose order on a file without actually rearranging the file. Indexing works in any direction.

CHAPTER 2

Requirement Analysis and Design

2.1 Domain Understanding

The main object of the project is to index all the key words of the document in a separate file and perform operations like finding and replacing. The outcome of this project is to ease the user for finding and replacing words with a friendly, understandable GUI. To process the document, the user is asked for the input to search and replace a word in the text file. When the user modifies the word in the text file all the possible position of the word gets replaced in the text file.

The operations done in this project are:

1. Indexing (each word of the input text file is indexed and the word with the index is stored in a separate output file)
2. Searching (the user inputs a word he wants to search in the document and that word is searched using the index and the lines containing that word are displayed)
3. Modification (the user inputs a word that he wants to change and enters the modified word. so all the possible occurrences of the word gets replaced by the modified word in the document).
4. Annotation (the user inputs a word for which the meaning is obtained and displayed as the output).

2.2 Classification of Requirements

2.2.1 System Requirements

Operating System: Windows

Spyder installed

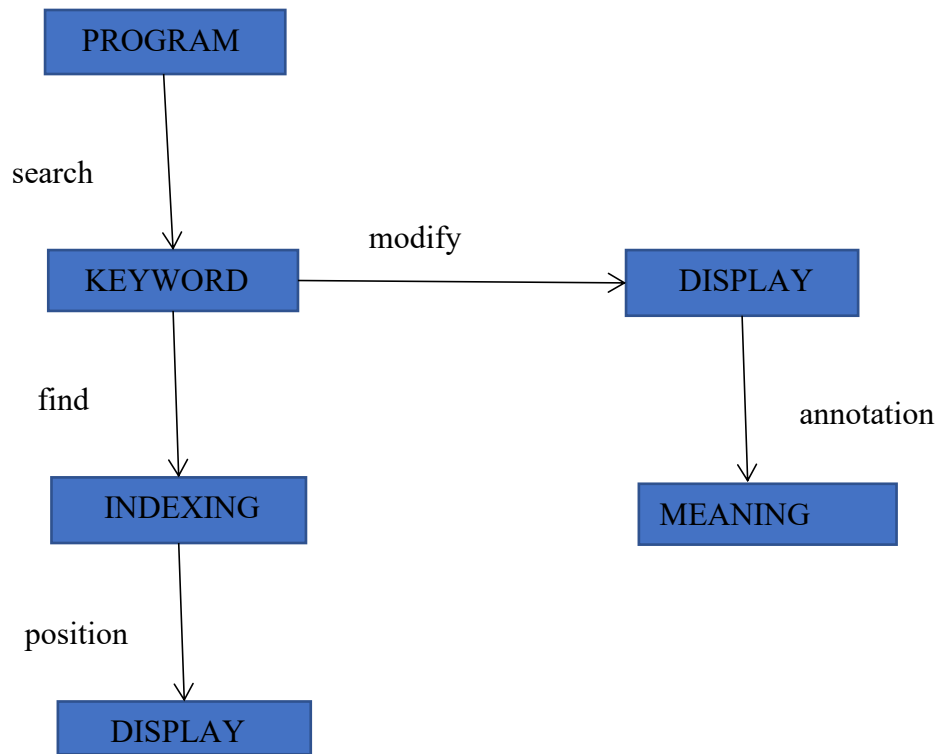
2.2.2 Software and Hardware Requirements

Programming Language: Python

2.3 System Analysis

When the program is executed, it primarily asks the user to operate the operation presented that is to find. When the user chooses the option to find, then it asks for the user to input the file in which he wants to “find” the word in. After the file is obtained, the entire file is indexed and the index values are maintained in a separate file. After the completion of indexing the user is prompted for the word he wants to search for. The user, if the word is present, is displayed with the number of times the word has been repeated and also in which line and word is found.

2.4:BLOCK DIAGRAM OF THE OPERATIONS PERFORMED



CHAPTER 3

IMPLEMENTATION

3.1 Indexing

Algorithm for indexing:

Step 1: Two files are opened, the file containing the data and the index file.

Step 2: Count is incremented to 0, and all the lines in the file are read.

Step 3: for line in s.split(" "):

```
Stri=line[:]
Index.write(stri+ " ")
c=str(count)
Index.write(c + "\n")
Count=len(line) + count
```

Step 4: the indexes of the file will be stored in the index file.

```
8
9 import time
10 print("do you want to c indexing?y/n")
11 q=input()
12 start1=time.time()
13 if(q=='y'):
14     l=input("enter word")
15     with open(r"C:\Users\JIT\Desktop\fsmini.txt","r") as pack:
16         with open(r"C:\Users\JIT\Desktop\index2.txt","w") as index:
17             count=0
18             f=pack.read()
19             s=f.replace("\n"," ")
20             for line in s.split(" "):
21                 stri=line[:]
22                 index.write(stri+ " ")
23                 c=str(count)
24                 index.write(c + "\n")
25                 count=len(line) + count
26 end1=time.time()
27 print("indexing complete in:",end1-start1,'milliseconds')
28
29 print("indexing completed")
30
```

Fig 3.1.1

This function is mainly used to get the indexes of the document for which the processing operations are performed. In this project implementation of simple indexing is done so that the program generates the index file which contains the index of the words present in that text file. This helps us to know the index of any word and the operations like searching can be done on the basis of indexing. Which helps the time efficiency of the function. In this project the time took to do indexing is also calculated by taking its start and end time. On the coding basis time is a built-in module which can be imported directly during the program. From this time module we can plot the graph of the time consumed by the program to run and check the efficiency. Indexing can be primary index or secondary index, since the data set in this project used is a large text file simple index is more appropriate. Here we can see code where the document is indexed using a simple indexing algorithm. Each word is taken into for and indexed in separate output file.

3.2 Searching

Algorithm for searching:

Step 1: The File containing the large text document is opened.

Step 2: A variable is declared to take the input of the word to be searched from the user.

Step 3: An empty string is declared and count is initilized to 1.

Step 4: All the lines in the file are read and split using the split function

```
s=f1.readline()
```

```
L=s.split()
```

Step 5: using while loop we loop through everyline in the file,and print the line number and the line if the word is present.

```
While(s):
```

```
    If word in l:
```

```
        Print("line number:",count,":",s)
```

```
        Count+=1
```

```
30
31 print("do u want to search?y/n")
32 question1=input()
33 start2=time.time()
34 if(question1=='y'):
35     f1=open(r"C:\Users\JIT\Desktop\fsmini.txt",encoding="utf8", errors='ignore')
36     word=input("enter the word to be searched:")
37     s=""
38     count=1
39     s=f1.readline()
40     l=s.split()
41     print(l)
42     while(s):
43         s=f1.readline()
44         l=s.split()
45         if word in l:
46             print("line number:",count,":",s)
47             count+=1
48     end2=time.time()
49     print("searching completed in:" , end2-start2,'milliseconds')
50
51 if word not in l:
52     print("word not present")
```

Fig :3.2.1

Searching is one of the basic operations that can be performed on a document. In this project, according to the algorithm, firstly the large text file has to be opened and all the lines in the file must be read. To make searching simple the words in the file are split using python built in method split. this helps the system to find the word in the text file much faster, the user inputs the word to be searched therefore a variable is declared to store the input, if the word is present in the file it prints the line number to which the word is present and the whole line in which the word is present. In this project time analysis is give importance so as to fine the efficiency of the program. the time starts when the input word is given and stops after displaying the required result. But if in case the input word is not present in the file it calculates the time taken to search the entire file and prints the message given by the programmer, in this case it is word not found. Along with

time taken to search. There are many ways in implementing searching on a file. This is one of the efficient ways to find the word in the large text file. In this project, the user is asked for the word to be searched in the text document, and when the program is executed, it searches and displays the line where the word appears along with the line number where it is present.

3.3 Modification

Algorithm for Modification:

Step 1: The file is opened.

Step 2: Two variables are declared to take the input of the word to be modified and the modified word.

Step 3: The value is replaced and stored in a variable

Replaced_value=data.replace(f2,f3)

Step 4: The file writes the modified word in all the occurrences of the word in the file

Step 5: The file is closed.

```
question=input("do u want to modify y/n:")

▼ if question=='y':
    f1=open(r"C:\Users\JIT\Desktop\fsmini.txt",encoding="utf8", errors='ignore')
    data=f1.read()
    linenumber=int(input("enter the line number to be modified"))
    f2=input("enter the word:")
    f3=input("enter the modified word:")

    replaced_value=data.replace(f2,f3)

    f1=open(r"C:\Users\JIT\Desktop\fsmini.txt","wt")
    f1.write(replaced_value)
    f1.close()
    print("modified successfully")
▼ else:
    print("end of program")
```

Fig:3.3.1

Modification is another operation that can be performed on a document. In this operation, one particular word in the text file can be replaced by another word. Using this, all the occurrences of the word in the file get replaced by the new word entered by the user. Programming in Python has made this function easier by using the replace built-in function. Mainly, the file is opened and a variable is declared to read the word, and another variable is declared to read the modified word. This word can be any user-defined string. Then, the program rewrites the places where the word has to be modified, and then prints the message. The difficulty faced was to print the line which was modified. In this project, I tried using linecache to print the line of the modified word, since it was a large text file, it failed to print the line and could not print all the lines which got modified. The future development of the project is to do the same, and improve the algorithm of the modification program. For now, in this program, the user is asked to modify any word in the text file, and the word gets replaced in all possible occurrences in the file.

3.4 Annotation

Algorithm:

Step 1: from PyDictionary import PyDictionary

Step 2: dictionary=PyDictionary()

Step 3:a=input("enter the word:")

Step 4:print(dictionary.meaning(a))

```
8
9  from PyDictionary import PyDictionary
10 dictionary=PyDictionary()
11 a=input("enter the word")
12 print(dictionary.synonym(a))
```

Fig:3.4.1

Annotation is one of the special function which can be performed on a document. In this project,the above algorithm describes of how annotation is implemented. For this special function installation of built in special modules call PyDictionary is necessary. Which is installed by typing the command on the terminal.In this modules meanings,synonyms,antonyms,translations etc likewise many things comes built in and be used directly by importing it. This module uses python requests,beautifulsoup4 and goslate as dependencies.PyDictionary can be utilized in 2 ways,either by creating a dictionary instance which can take words as arguments or by creating a dictionary.PyDictionary is installed from the command **pip install PyDictionary** Then it is imported in the file,using which the meanings of any word can be obtained.

Algorithm (another method):

Step 1:import webbrowser

Step 2:search_terms=['File Structures']

Srep 3: for term in search_terms:

Url="https://www.google.com.tr/search?q={}".format(term)

Webbrowser.open_new_tab(url)

```
8
9  import webbrowser
10 search_terms=['File Structures']
11 for term in search_terms:
12     url="https://www.google.com.tr/search?q={}".format(term)
13     webbrowser.open_new_tab(url)
```

Fig 3.4.2

This is another method of doing annotation by importing webbrowser. This method directly links the google webbrowser. The advantage of using this method is that it shows more information when compared to the previous method as it is linked to the google webpage. N number of information can be obtained using this method.

CHAPTER 4
RESULTS AND SNAPSHOTS

Results of Indexing:

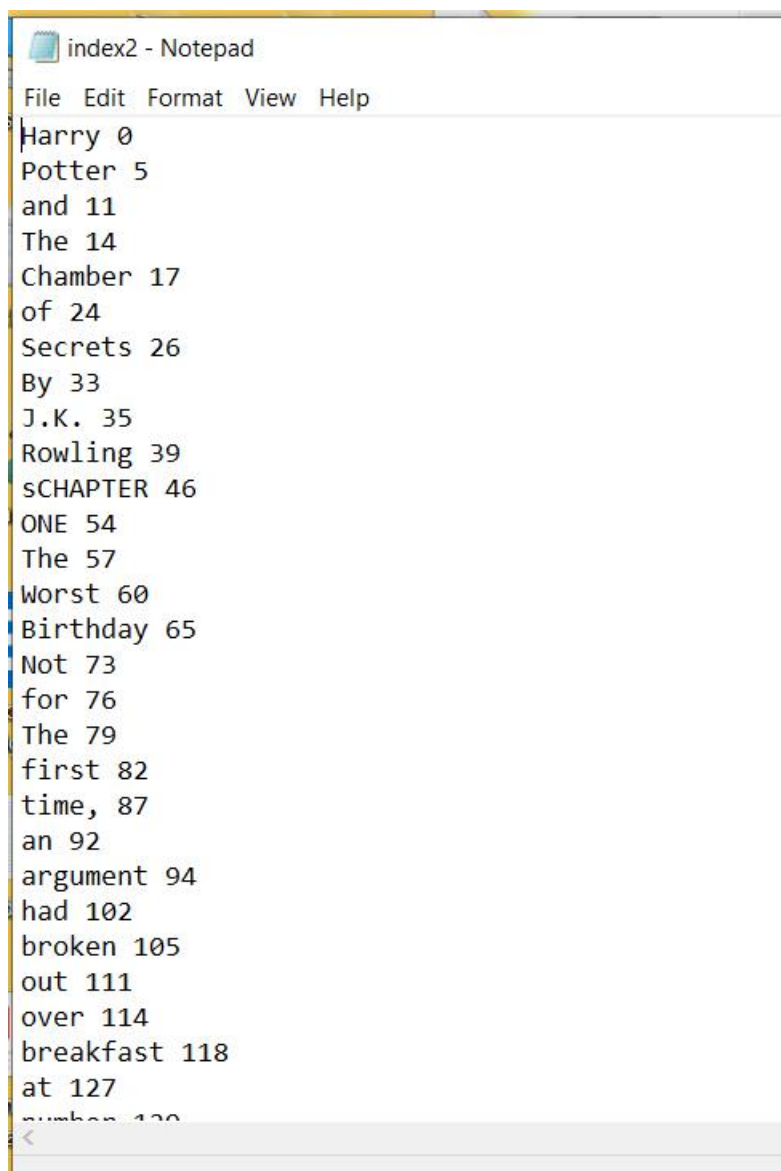
```
In [8]: runfile('C:/Users/JIT/untitled6.py', wdir='C:/Users/JIT')
do you want to c indexing?y/n

y

enter wordHarry
indexing complete in: 4.531103134155273 milliseconds
indexing completed
do u want to search?y/n

|
```

Fig 4.1: The user is asked whether he wants to do indexing if yes the file gets indexed and the time taken for the completion of the indexing is calculated and displayed.



The screenshot shows a Notepad window titled "index2 - Notepad" with a menu bar (File, Edit, Format, View, Help). The text content is a list of words followed by their indexed positions, separated by a space. The list includes: Harry 0, Potter 5, and 11, The 14, Chamber 17, of 24, Secrets 26, By 33, J.K. 35, Rowling 39, sCHAPTER 46, ONE 54, The 57, Worst 60, Birthday 65, Not 73, for 76, The 79, first 82, time, 87, an 92, argument 94, had 102, broken 105, out 111, over 114, breakfast 118, at 127, number 128, and a partially visible line starting with "<".

Fig 4.1.1:all the words present in the document are indexed and stored inside a separate file.

Results of Searching:

```
do u want to search?y/n
y
enter the word to be searched:Harry|
```

Fig 4.2: The user is asked to input the word to be searched.

```
Console 1/A
knocked into their plates of trifle, or his and Kongs four hundred points for Gryffindor securing the
House Cup for The second year running, or Professor McGonagall standing up to tell Them all that The
exams had been canceled as a school treat (Oh, no! said Hermione), or Dumbledore announcing that,
unfortunately, Professor Lockhart would be unable to return next year, owing to The fact that he
needed to go away and get his memory back. Quite a few of The teachers joined in The cheering that
greeted this news.

line number: 934 : Too soon, it was time for The journey home on The Hogwarts Express. Harry, Ron,
Hermione, Fred, George, and Ginny got a compartment to Themselves. They made The most of The last few
hours in which They were allowed to do magic before The holidays. They played Exploding Snap, set off
The very last of Fred and Georges Filibuster fireworks, and practiced disarming each oTher by magic.
Harry was getting very good at it.

line number: 935 : They were almost at Kings Cross when Harry remembered something.

line number: 936 : Harry pulled out his quill and a bit of parchment and turned to Ron and Hermione.

searching completed in: 4.508213520050049 milliseconds
```

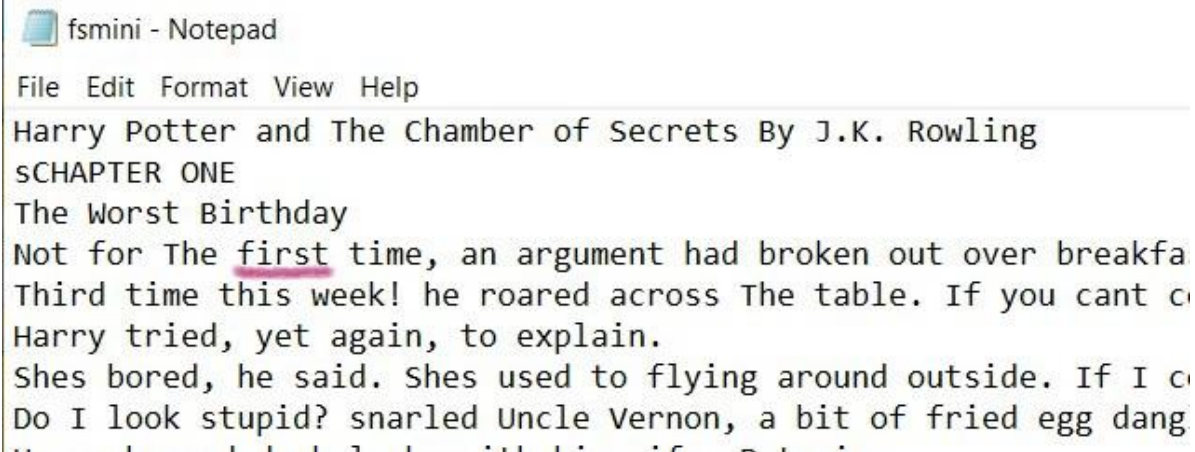
Fig 4.2.1: The searching operation is performed and all the lines where the word is present is displayed along with the line number. Time taken to search a particular word is calculated.

```
do u want to search?y/n
y
enter the word to be searched:HAPPY
['Harry', 'Potter', 'and', 'The', 'Chamber', 'of', 'Secrets', 'By', 'J.K.', 'Rowling']
searching completed in: 5.3867292404174805 milliseconds
word not present

do u want to modify y/n:|
```

Fig 4.2.2: The word is searched in the file, if the word is not present in the file then it prints word not present.

Results of Modification:



fsmini - Notepad

File Edit Format View Help

Harry Potter and The Chamber of Secrets By J.K. Rowling

CHAPTER ONE

The Worst Birthday

Not for The first time, an argument had broken out over breakfast at number four, Privet Drive. Mr. Vernon Dursley

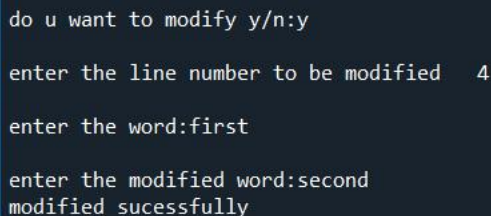
Third time this week! he roared across The table. If you cant control that owl, itll have to go!

Harry tried, yet again, to explain.

Shes bored, he said. Shes used to flying around outside. If I could just let her out at night -

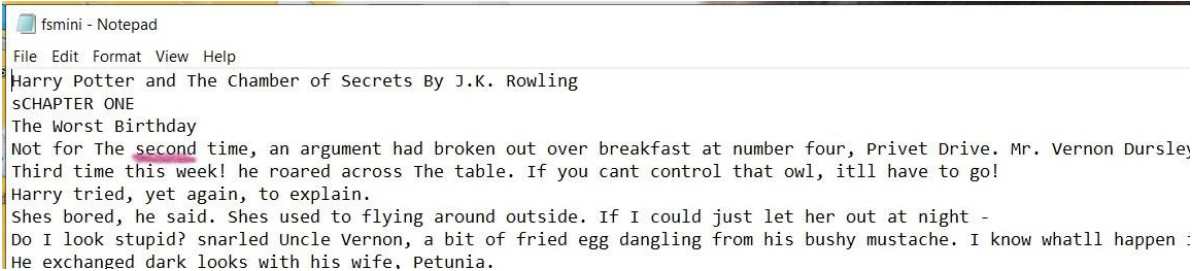
Do I look stupid? snarled Uncle Vernon, a bit of fried egg dangling from his bushy mustache. I know whatll happen :

Fig 4.3: This snapshot is of the input file, and the underlined word is the word which is getting modified.



```
do u want to modify y/n:y
enter the line number to be modified 4
enter the word:first
enter the modified word:second
modified sucessfully
```

Fig 4.3.1: when the program is executed, it asks the user to enter the word and then the modified word, if the process is completed it prints modified successfully.



fsmini - Notepad

File Edit Format View Help

Harry Potter and The Chamber of Secrets By J.K. Rowling

CHAPTER ONE

The Worst Birthday

Not for The second time, an argument had broken out over breakfast at number four, Privet Drive. Mr. Vernon Dursley

Third time this week! he roared across The table. If you cant control that owl, itll have to go!

Harry tried, yet again, to explain.

Shes bored, he said. Shes used to flying around outside. If I could just let her out at night -

Do I look stupid? snarled Uncle Vernon, a bit of fried egg dangling from his bushy mustache. I know whatll happen :

He exchanged dark looks with his wife, Petunia.

Fig 4.3.2: The output is something like this, the underlined word as mentioned above got modified to the user entered word. The drawback of this algorithm is that it cannot print all the lines which are modified in the file as there are thousands of lines in the text file.

Results of Annotation:

```
In [3]: runfile('C:/Users/JIT/untitled10.py', wdir='C:/Users/JIT')

enter the word beautiful
{'Adjective': ['delighting the senses or exciting intellectual or emotional admiration', '(of weather)']}

In [4]: |
```

Fig 4.4:The user enters the word and the meaning of the word is displayed.

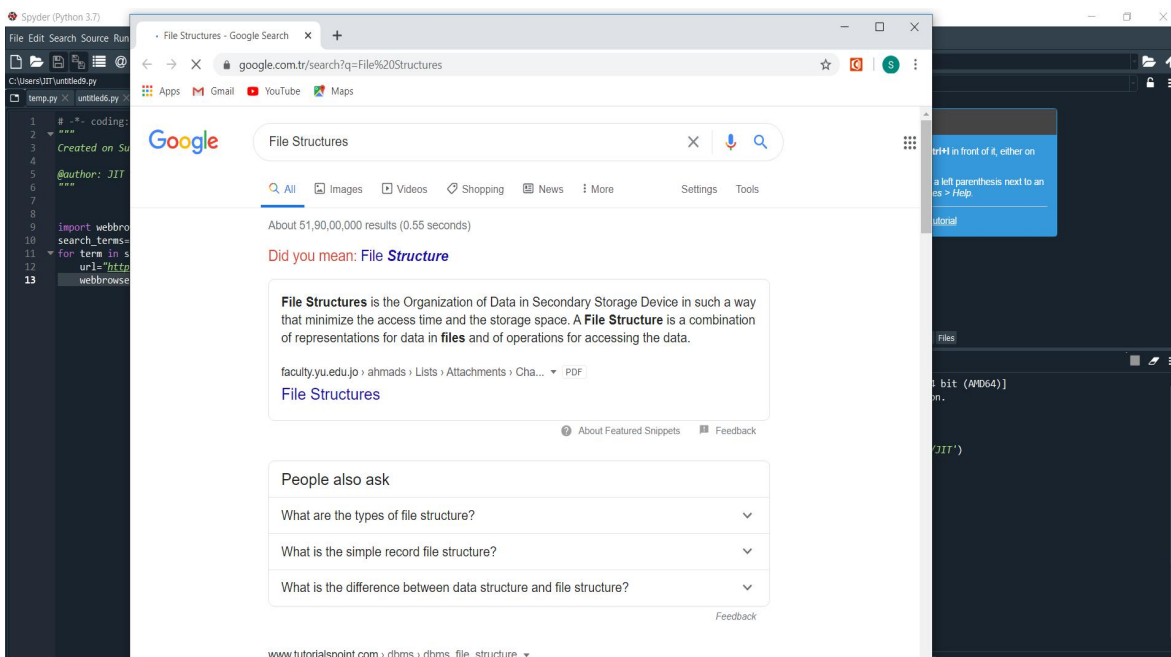


Fig 4.4.1:This is the output of the second algorithm discussed in the implementation chapter, The description of the input word is obtained when the program is run. The description of the input word is displayed through a Google webpage. Any number of information can be obtained using this method.

4.5 Time Analysis of Indexing and Searching

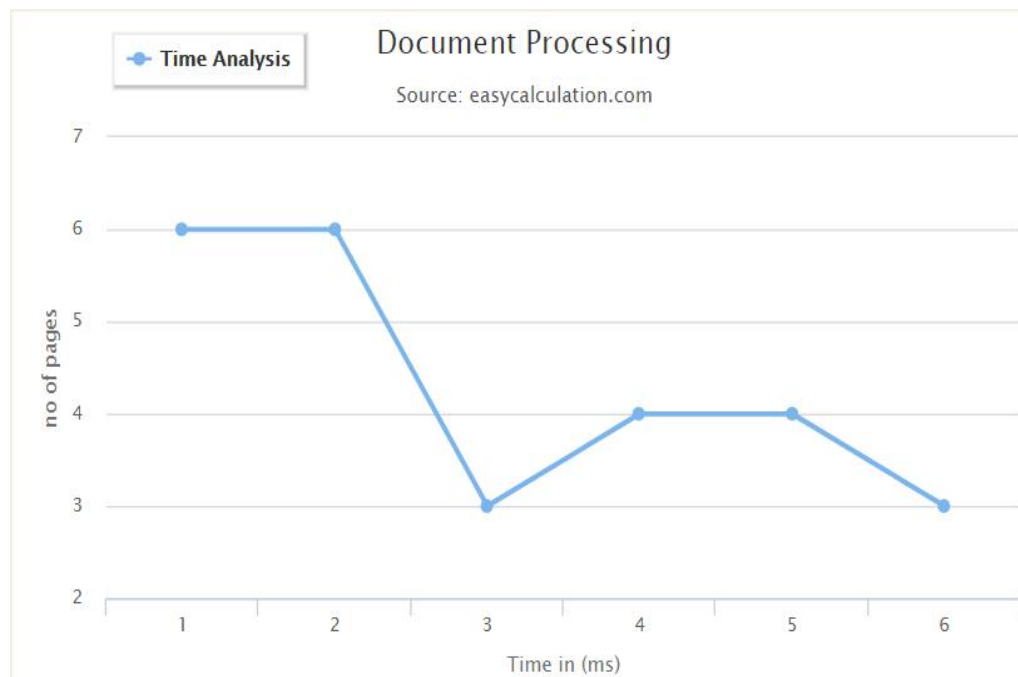


Fig 4.5: The above graph represents the time analysis of search function. where the graph is plotted by taking various points (by varying number of lines) against time taken to do search function in milliseconds.

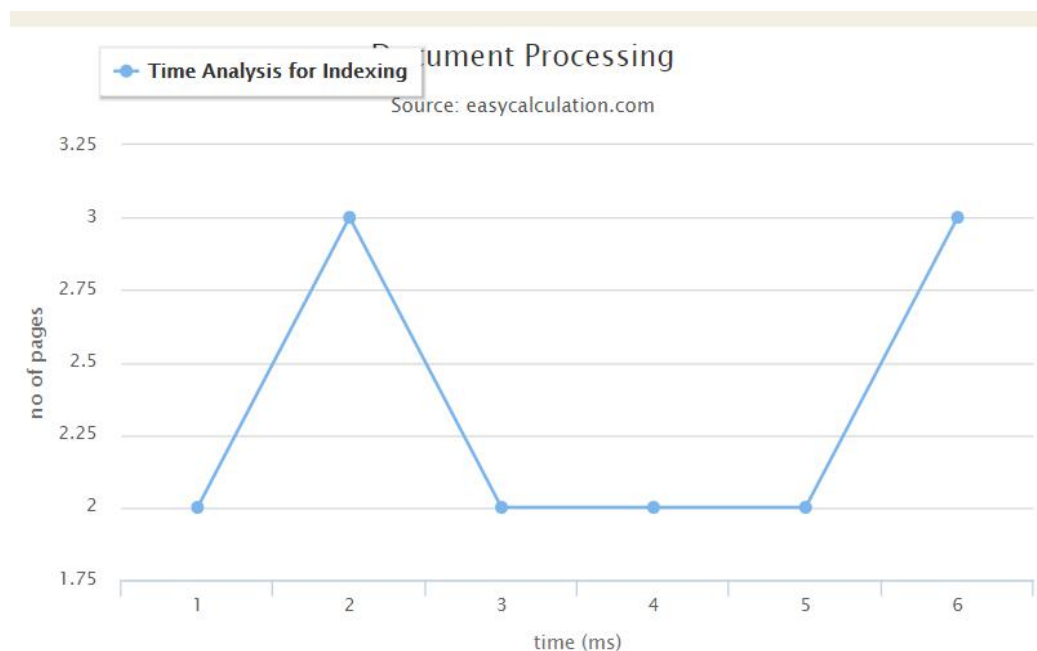


Fig 4.5.1: The above graph represents the time analysis of index function. where the graph is plotted by taking various points (by varying number of lines) against time taken to do indexing of the text in the file in milliseconds.

Number of lines	Time taken for Indexing(milliseconds)	Time taken for Searching(milliseconds)
500	2.81	6.07
1000	3.56	6.66
1500	2.18	3.39
2000	2.76	4.43
2500	2.80	4.94
3000	3.11	3.73

Fig 4.5.2: The above table shows the amount of time required to perform indexing and searching for different number of lines.

Conclusion

As the project was progressive in nature, I learnt a lot about Python and its modules. It also helped me understand various concepts of file structures. My learning curve and also in coding has increased with this mini project. The concept of indexing and its implementation has been more clear when practically implemented. I hereby conclude this mini project “Document Processing using keywords ” successfully with the best of my ability.

FUTURE WORK

- 1.Implementing better searching and indexing mechanisms
- 2.creating a smoother GUI for better understanding and output display

Reference:

1. File Structures-An Object Oriented Approach with C++
Michael J. Folk, Bill Zoellick, Greg Riccardi
2. <https://www.computerhope.com/issues/ch001721.htm>
3. <https://stackoverflow.com/questions/4940032/how-to-search-for-a-string-in-text-files>
- 4 File Structures Using C++
K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj .
5. <https://www.easycalculation.com/graphs/line-graph.php>