

**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 Lab [17ISL68]

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

Submitted by

SHASHANK S [1JT17IS038]

Under the guidance

Mr. Vadiraja A

Assistant Professor, Dept of ISE



**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 **SHASHANK S [1JT17IS038]** bona fide student of **Jyothy Institute of Technology**, in partial fulfilment for the award of **Bachelor of Engineering in Information Science and Engineering** department of the **Visvesvaraya Technological University, Belagavi** during the year **2020-2021**. 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.

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

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

External Viva Examiner

- 1.
- 2.

Signature with Date:

ACKNOWLEDGEMENT

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

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

I would like to thank **Dr. Harshvardhan 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 our friends who have helped us directly or indirectly in this project.

SHASHANK.S[1JT17IS038]

ABSTRACT

This project has been created using Spyder, with a Windows platform. The project title is “**DOCUMENT PROCESSING USING KEYWORDS**” talks about how we process a text file and index the data and also use it to retrieve certain details in efficient way.

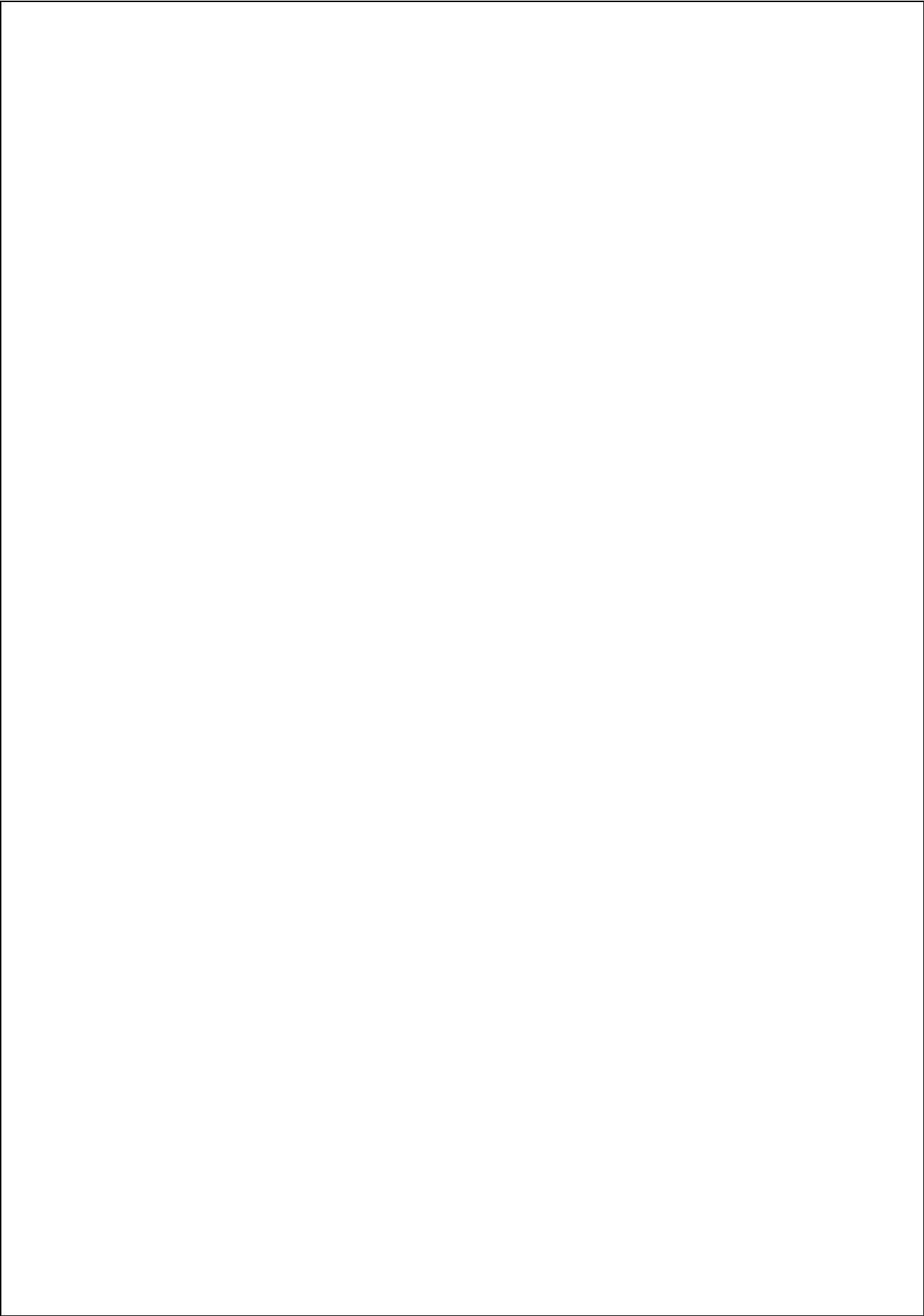
The purpose of this project is to reduce the time it takes to search keywords in a file by the help of computerized equipments and full-fledged computer software fulfilling their requirements, so that their valuable data/information can be retrieved faster with easily available and easy to work with.

It can assist the user to come up with a better document by looking at the keywords the system generates by processing the user’s document.

.

Table of Content

SL No	Description	Page No
1	INTRODUCTION	1-5
	1.1 Introduction to File Structures	1
	1.2 File System	2
	1.3 Introduction to Python	3
	1.4 Document Processing Introduction to Python	4
	1.5 Indexing	5
2	REQUIREMENT ANALYSIS AND DESIGN	6-8
	2.1 Domain Understanding	6
	2.2 Classification of Requirements	6
	2.3 System Analysis	7
3	IMPLEMENTATION	9-10
	3.1 Indexing	9
	3.2 Searching	9
	3.3 Modifying	10
4	RESULTS AND SNAPSHOTS	11-13
5	ANALYSIS	14-16
6	FUTURE SCOPE	17
7	CONCLUSIONS	17
8	REFERENCES	18



1. INTRODUCTION

1.1 Introduction to File Structures

In simple terms, a file is a collection of data stored on mass storage (e.g., disk or tape).

But there is one important distinction that must be made at the outset when discussing file structures. And that is the difference between the logical and physical organization of the data.

On the whole a 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. It is this logical aspect that we will concentrate on. The physical organization is much more concerned with optimizing the use of the storage medium when a particular logical structure is stored on, or in it. Typically for every unit of physical store there will be a number of units of the logical structure (probably records) to be stored in it.

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.

Like all subjects in computer science the terminology of file structures has evolved higgledy-piggledy without much concern for consistency, ambiguity, or whether it was possible to make the kind of distinctions that were important.

It was only much later that the need for a well-defined, unambiguous language to describe file structures became apparent. In particular, there arose a need to communicate ideas about file structures without getting bogged down by hardware considerations.

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. Thus, one might say “I have two file systems” meaning one has two partitions on which one stores files, or that one is using the “extended file system”, meaning 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 superblock, inode, data block, directory block and indirection block. The superblock 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 (byte code-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.

An excellent way to see how Python code works is to run the Python interpreter and type code right into it. If you ever have a question like, "What happens if I add an int to a list?" Just typing it into the Python interpreter is a fast and likely the best way to see what happens.

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

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).

IMPROVEMENTS IN DOCUMENT PROCESSING - An economic appraisal of the problem of improving business data processing involves balancing the costs of making a change and using the new technique against the benefits derived.

Important strides in mechanizing data processing are being made through the use of document reading devices. Character recognition machines are available from several manufacturers to read either one or perhaps a variety of type styles.

Improvement in document processing should, in the rational world of economics, be determined by balancing:

- 1) The cost of operating the system utilizing the new technology versus the old.
- 2) Cost of transition from one to another.
- 3) The advantages derivable from the new technology.

Here we are using the document processing in such a way that we are searching for specific words in a certain electronic document. We can find and also replace the words we have found. It is done with the help of a concept called, Indexing. Indexing is explained in detail in the next sub-section.

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.

1.5 Indexing

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.

One of the advantages of indexed files is that we can have more than one index, each with a different key. For example, an employee file can be retrieved based on either social security number or last name. This type of indexed file is usually called an inverted file.

This is the main aspect in this mini project, inverted indexing. It is designed to allow very fast full-text searches. An inverted index consists of a list of all the unique words that appear in any document, and for each word, a list of the documents in which it appears.

To create an inverted index, we first split the content field of each document into separate words, create a sorted list of all the unique terms, and then list in which document each term appears. If we apply a naive similarity algorithm that just counts the number of matching terms, then we can say that the first document is a better match—is more relevant to our query—than the second document.

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 location of input is taken from the user. Main purpose of taking the location input is that the user can follow the operations for any file document. The operations done in this project are:

- Indexing (each word of the input text file is indexed and the word with the index is stored in a separate output file)
- 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)

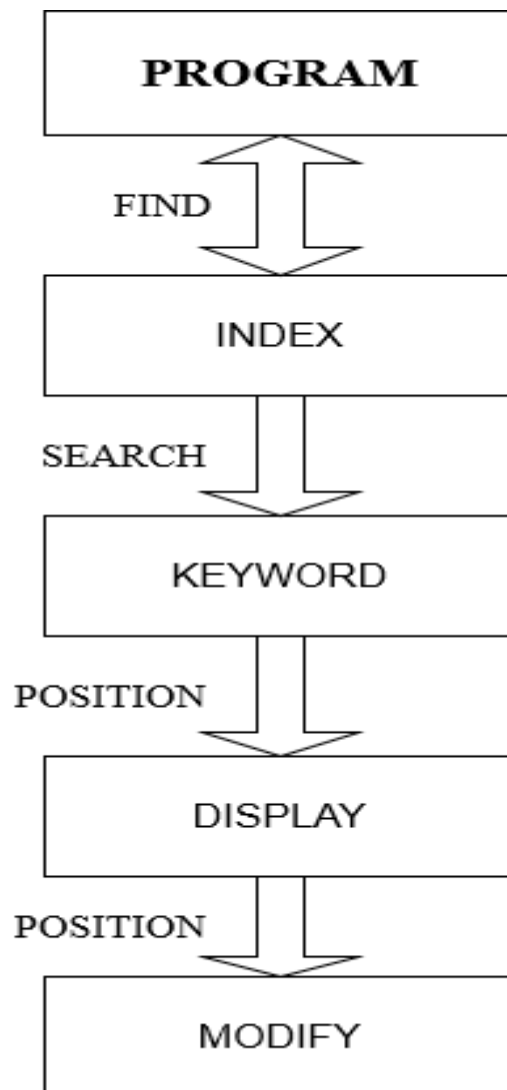
2.2 Classification of Requirements

1. Any Text editors (IDE preferred - Spyder, Pycharm etc.)
2. Any Windows/Mac/Linux distributions

2.2 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 the word is found.

Block diagram of the search operation performed in the project:



This block diagram represents the indexing, search and modify operations performed by the written program.

3. IMPLEMENTATION

3.1 Indexing

Algorithm:

Step1: Open an input document that has to be processed.

Step2: Calculates the number of lines in the document.

Step3: Open an empty document in write mode to store the indexes of the input document.

Step4: Read all the lines in the input document, split the words in it and returns the indexes of the respective words.

Step5: The words and its indexes in written into the output file.

In the above algorithm, firstly it opens the document and the total number of lines in it is calculated and then the document is read again, an empty document is opened in write mode and starts indexing using simple indexing algorithm. Each word in the document is splitted and returns the respective indexes the split words are stored in an array and writes it into a separate output file, which is later used in the program to perform the search operation to find the specific words.

3.2 Searching

Algorithm:

Step1: Open an input document that has to be processed.

Step2: Open the index file and takes the user input for the word to be searched.

Step3: Searches the given word in both input document and index file.

Step4: If the word is found, returns the entire line consisting the searched word with its index.

Step5: Else it returns word not found.

In the above algorithm, firstly it opens the input document and the indexes document here we take the user input for the word to be searched and search using linear search algorithm where it compares with each word using the positions in the index file which we have stored previously if the word exists in the document and returns the line consisting the searched word and its index else returns word not found.

3.2 Modifying

Algorithm:

Step1: Take user input whether the word need to be modified or not.

Step2: If Yes, Open an input document that has to be processed.

Step3: Prompts the user to enter the word to be replaced and replace with:

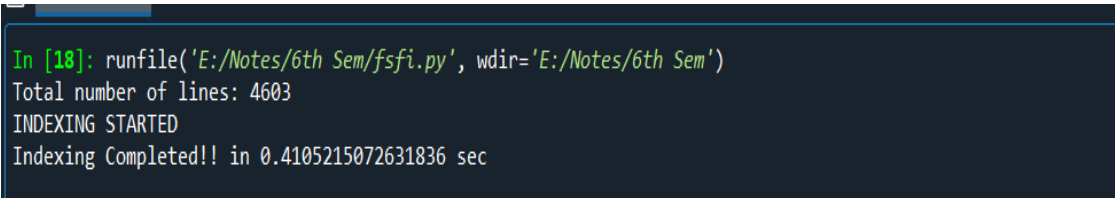
Step4: If the word exists in the document, replace the words.

Step5: Else the program terminates and the file is closed.

Here we take user input, whether the word searched is to be modified or not. On the user's input the above algorithm modifies the word to be replaced with the other word if the word is found else it terminates and closes the file.

4. RESULTS

Indexing:

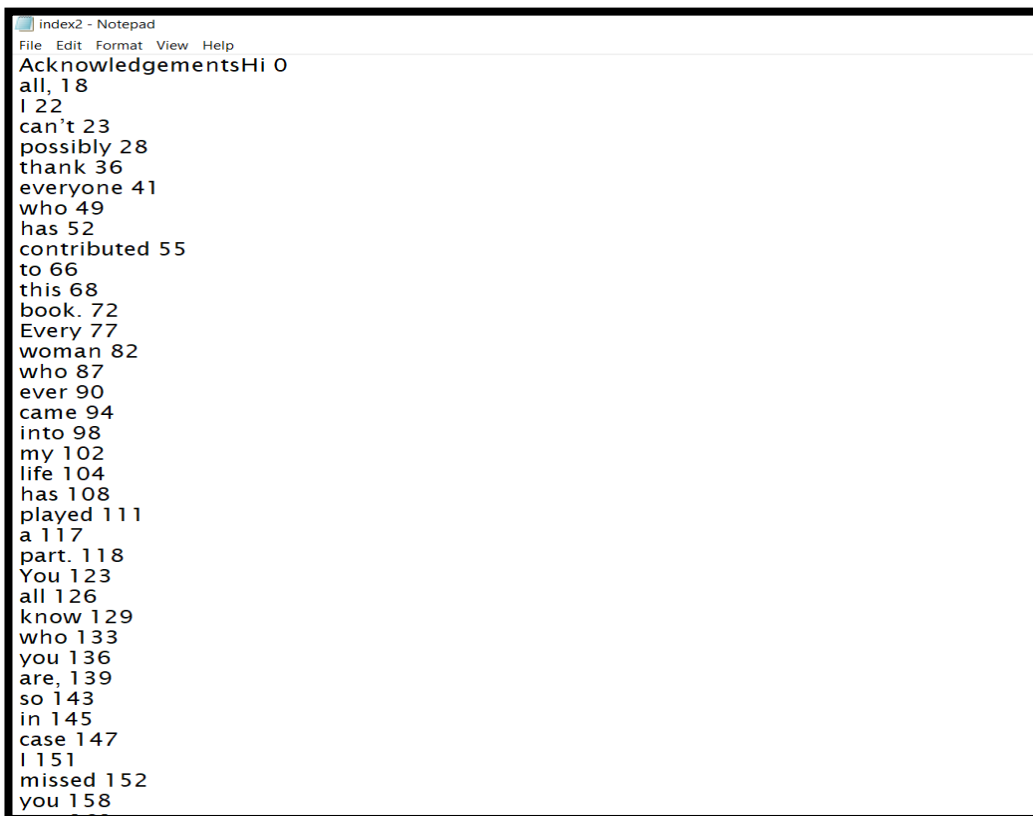


```
In [18]: runfile('E:/Notes/6th Sem/fsfi.py', wdir='E:/Notes/6th Sem')
Total number of lines: 4603
INDEXING STARTED
Indexing Completed!! in 0.4105215072631836 sec
```

Fig 4.1

In the above **Fig 4.1** output first it returns the total number of lines in the input document.

Starts indexing and stores in the output file and also returns the time taken to complete indexing operation.



```
index2 - Notepad
File Edit Format View Help
AcknowledgementsHi 0
all, 18
I 22
can't 23
possibly 28
thank 36
everyone 41
who 49
has 52
contributed 55
to 66
this 68
book. 72
Every 77
woman 82
who 87
ever 90
came 94
into 98
my 102
life 104
has 108
played 111
a 117
part. 118
You 123
all 126
know 129
who 133
you 136
are, 139
so 143
in 145
case 147
I 151
missed 152
you 158
```

Fig 4.2

Output file with the indexes of each word.

Searching:

```
In [21]: runfile('E:/Notes/6th Sem/fsfi.py', wdir='E:/Notes/6th Sem')
Total number of lines: 4603
INDEXING STARTED
Indexing Completed!! in 0.3650078773498535 sec

Enter the word to be searched:Neel
```

Fig 4.3

In above **Fig 4.3** is the first step of searching operation where it takes user input, which word to be searched.

```
The line is here in index fileNeel 354987
away. I can't hold your hand here, can I? Even though you do like to spoon me at night, isn't it?' Neel
'Life isn't lived "out of this world", Neel. Life is lived in this world.' To be fair, even I didn't know why I felt
the problem, Neel. I can't even discuss us with anyone. Other girls discuss their boyfriends with their
washing my face. Neel looked at me with concern. 'Sorry,' I said. Why am I saying sorry to him? 'You don't
My plate of vegetable dumplings had gone cold. One of the waiters replaced it. Neel and I ate in silence. 'I
love you,' Neel said. I shrugged. 'So?' I said. 'Kind of irrelevant, isn't it?' 'What do you want, Radha?'
Neel said. I kept silent. 'A future? I am twenty years older,' he said. 'You said age doesn't matter in love.
you happy with just what we have?' Neel said. He seemed to be genuinely confused. 'Would you be? If you
soon?' 'Eventually I do want to, Neel. How could you think I won't? I want marriage, kids, family.' 'Really?'
'Really, Radha?' Neel said. He looked at me gobsmacked, as if I had revealed my secret
supposed to mean?' I said, my voice ice-cold. 'Nothing. Let's talk later. I'll ask for the bill,' Neel said and
'No, Neel. Not tonight,' I said. We were lying tangled up in my bed. He nuzzled the side of my neck. It
murmured my name in a persuasive voice. 'No, Neel,' I said. 'I am tired.' He had come to my apartment
on my side to face him. I lifted myself up to rest on my elbow. 'Really, Neel?' I said. 'What?' Neel said. 'I
over-anxious these days. Is it that time of the month?' He tried to put his arms around me. 'Shut up, Neel,'
shaken up. To really think about what I want.' Neel sat up in bed. 'Sorry to say this, but you aren't thinking
clearly. If I go by what you said in Dragon-I,' Neel
```

Fig 4.4


Next **Fig 4.4** is the output where it displays the indexes and line where the searched word exists.

```
as long as you fly beneath me, is it?' He looked down. I turned to Neel. 'Neel, you loved me as the flying
bird. You wanted me to fly higher and higher.' 'Of course,' Neel said. 'But you know where you went
wrong?' 'Where?' Neel said. 'You didn't want me to have a nest.' Neel didn't have an answer. 'Neel?' I
said. 'I believe in equal rights. You know that, right?' Neel said. 'Did you realize that perhaps I did not want
his head as he also heard and tried to figure out what I said. 'Meaning?' Neel said. 'What do you want?
Radha,' Neel said. I took a sip of my lukewarm coffee. 'Ah, choose,' I said and sneered. 'Choice. The
career.' 'What do you mean? Isn't that how it should be?' Neel said.
or sit in the nest.'" 'I don't get it. Really,' Debu said. Even Neel looked confused. 'From a man's
judgement, but they want sex, right?' 'Yeah,' Neel said. I collected my thoughts before I spoke again. 'Let's
make,' Neel said. 'Exactly, Neel. It is indeed a ridiculous choice. Just as ridiculous as the choice given to
Neel had something to say. 'I get it. Women want everything. To have a lovely home and be a great
mother. To also have a chance to shine in their careers,' Neel said. 'Not all, maybe, but for many, yes,' I
strong need to have a family too. That is why I made a mess last time,' Neel said. 'I imposed my notion of
making the right choice with me, Radha,' Neel said. 'No, Neel, not with you either.' 'What?' he said. I
'But Radha. . .' Debu began. I placed a finger on my lips to shush him right there. 'And Neel, you are
work so hard in the gym for. Well, I won't be this young girl forever. I don't know what Neel Gupta will do
do appreciate that. And do fuck off,' I said. Debu and Neel looked at each other. They looked at me once
Completed Searching in: 16.03800368309021 sec
```

Fig 4.5

Finally in **Fig 4.5** it returns the time required for the completion of the search operation.

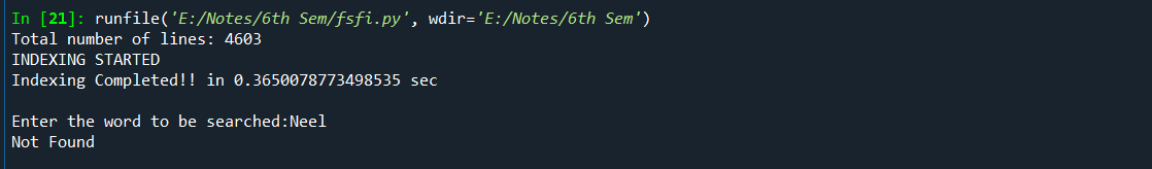
Modifying:



```
Do you want to modify Y/N:y  
Enter the word to be replaced:Neel  
Replace with:Nithin  
Replaced Successfully in 0.011976242065429688 sec
```

Fig 4.6

The above **Fig 4.6** is performed after the search operation the user is queried whether the word is to be replaced or not. If Yes then the word is replaced with the desired word. The time needed to perform modify operation is also returned.



```
In [21]: runfile('E:/Notes/6th Sem/fsfi.py', wdir='E:/Notes/6th Sem')  
Total number of lines: 4603  
INDEXING STARTED  
Indexing Completed!! in 0.3650078773498535 sec  
  
Enter the word to be searched:Neel  
Not Found
```

Fig 4.7

The above **Fig 4.7** is when the searched word doesn't exist in the given input document.

4. Analysis

Time Complexity:-

Time Vs Number of lines Graphs

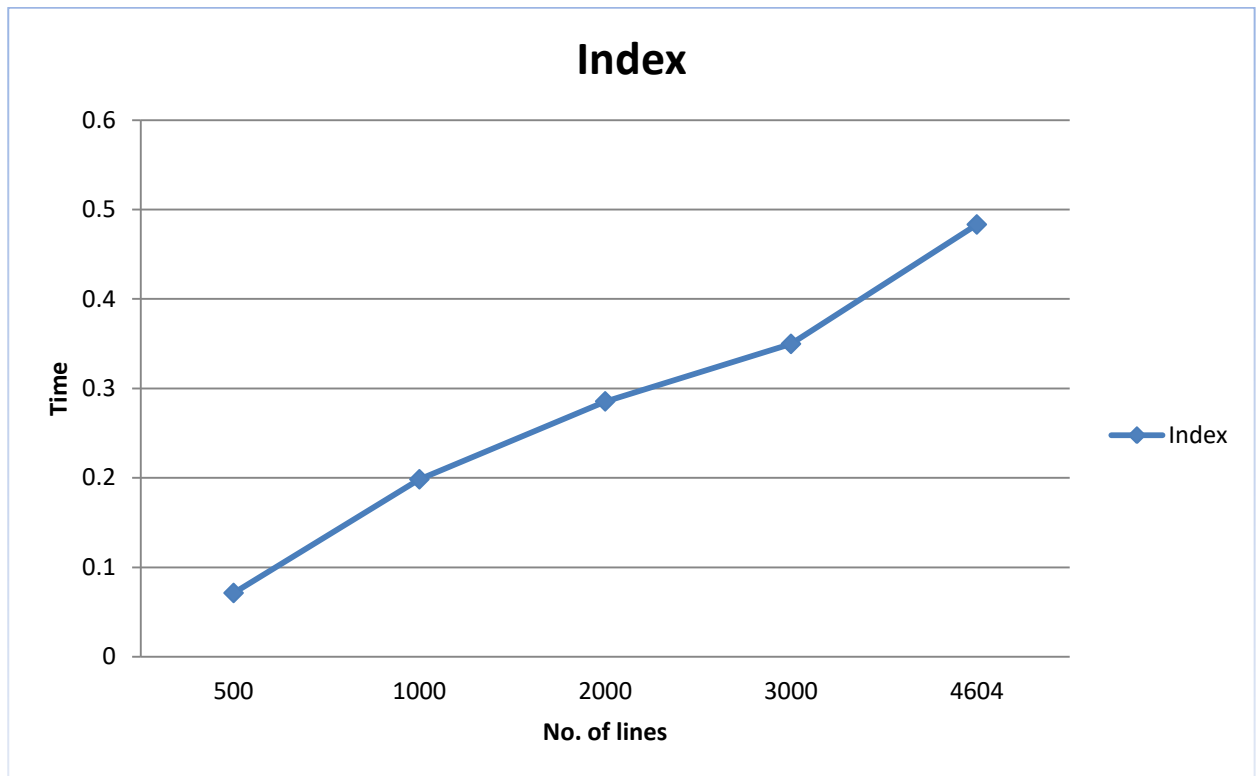


Fig 5.1-Time taken for Indexing

Number of lines	Time taken for Indexing(secs)
500	0.0712
1000	0.1983
2000	0.2854
3000	0.3497
4604	0.4832

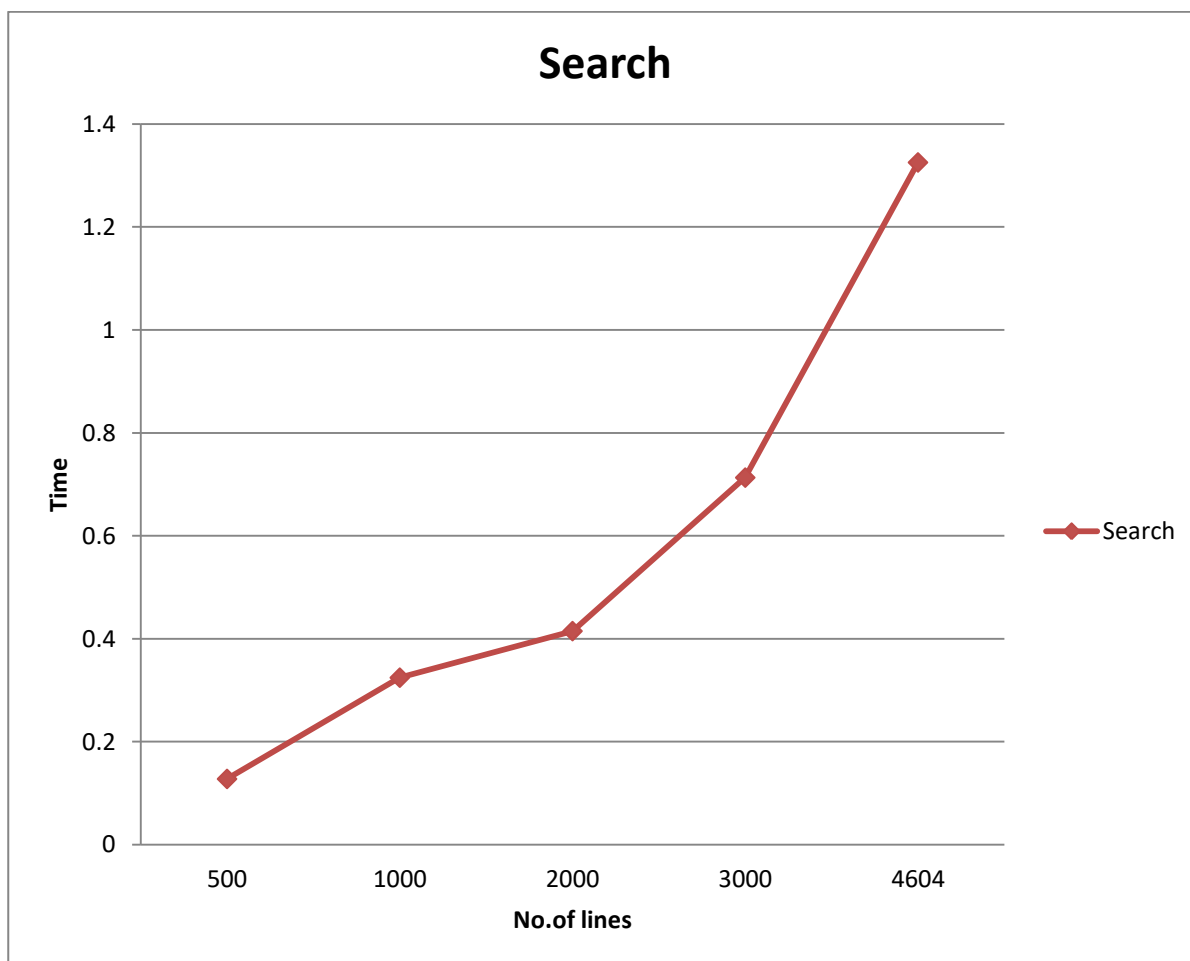


Fig 5.2-Time taken for Searching

Number of lines	Time taken for Searching(secs)
500	0.1279
1000	0.3246
2000	0.415
3000	0.7132
4604	1.3254

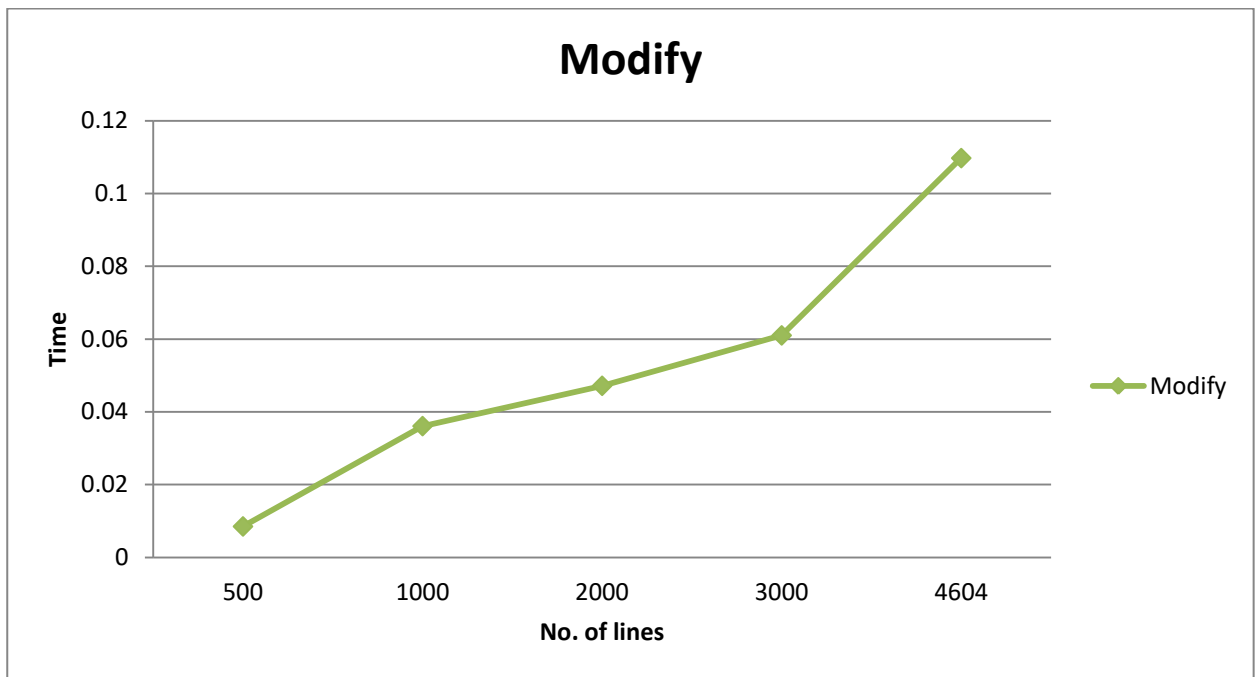


Fig 5.2-Time taken for Modifying

Number of lines	Time taken for Modifying(secs)
500	0.0085
1000	0.03607
2000	0.04715
3000	0.061
4604	0.10973

The above tables shows the amount of time required to perform indexing, searching and modifying for different number of lines.

Here in the Graph we can clearly see that when the number of lines increase, even the time taken for indexing, searching and modifying increases.

Time is the **Y-axis** and **Number of lines** is the **X-axis**. As we increase the input, time drastically increases almost exponentially.

5. FUTURE WORK

- ❖ We can add faster searching mechanisms.
- ❖ We can add a separate file for words which are searched often which will increase the probability of output in terms of speed.
- ❖ Showing where the words are in a separate window with the document.
- ❖ Creating a smoother GUI for better understanding and output display

6. 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 .It made me strong both in theoretical and practical aspects. My learning curve and also in coding has increased with this mini project. It was a very good learning experience in terms of time management and learning objectives.

I hereby conclude this mini project titled “**Document Processing Using Keywords**” successfully with the trust of my senses and to best of my ability.

7. REFERENCES

- [1] File Structures - An Object Oriented Approach with C++ Michael J. Folk, Bill Zoellick, Greg Riccardi
- [2] File Structures Using C++ K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj
- [3] <https://www.codesdope.com/python-while/>
- [4] https://www.cs.uct.ac.za/mit_notes/database/htmls/chp11.html
- [5] <https://stackoverflow.com/questions/4940032/how-to-search-for-a-string-in-text-files>
- [6] <https://www.javatpoint.com/python-strings>
- [7] https://www.w3schools.com/python/python_lists.asp