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

RAKSHITHA.M[1JT17IS033]

Under the guidance

Mr. Vadiraja A

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 Keyword"** carried out by **Rakshitha.M[1JT17IS033]** 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	**Dr. Harshvardhan Tiwari**
Guide,Asst Professor	Associate Professor and HOD
Dept of ISE	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 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, Associate Professor and Head of Information Science and Engineering Department for providing for 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, we would thank all our friends who have helped us directly or indirectly in this project.

RAKSHITHA.M[1JT17IS033]

ABSTRACT

A file structure is a combination of representation of data in files and operations for accessing the data. It allows applications to read, write and modify data. The project titled "**Document Processing Using Keyword**", where it helps to search keywords in a text file. In this process, the large text file is exposed to perform operations like search, modify and indexing where search helps to find keywords in a file, modify helps in altering the information and indexing helps to find the exact position of the words in the text file. The purpose of this project is to reduce the time consumed to perform these operations by the help of computerized equipments and full-fledged computer software fulfilling their requirements.

Table of Content

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 (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.

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 distribution

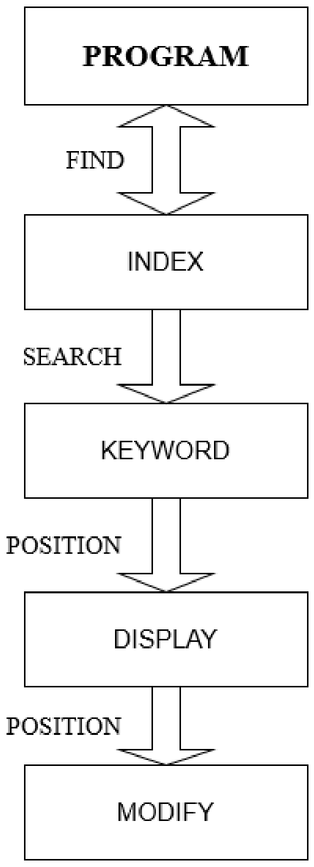
2.3 System Analysis

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 the word is found.

2.4 Block diagram of the search operation performed in the project



The block diagram representing the indexing, search and modifying operations performed in the program.

3. IMPLEMENTATION

3.1 Indexing

<u>Algorithm:</u>

<u>Step1</u>: Open an input document that has to be processed.

<u>Step2</u>: Calculates the number of lines in the documents.

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

<u>Step4</u>: Reads all the lines in the document and split the words in it and returns the indexes of the respective words.

<u>Step5</u>: The words and the indexes in the output file.

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

3.2 Searching

<u>Algorithm:</u>

<u>Step1</u>: Open an input document that has to be processed.

<u>Step2</u>: Opens the index file and takes the user input for the word to be searched.

<u>Step3</u>: Search the given word in both input document and index file.

<u>Step4</u>: If the word is found, returns the entire line consisting the searched word with its index

<u>Step5</u>: Else it returns search is unsuccessful

In the above algorithm firstly it opens the input document and the index document, here we take the user input for the word to be searched using linear search algorithm. If the word is found it returns the word with its position or else it will return the word is not found.

3.3 Modifying

<u>Algorithm</u>:

<u>Step1</u>: Takes user input whether the word need to be modified or not.

<u>Step2</u>: If yes, open the input document that has to be processed.

<u>Step3</u>: Prompts the user to enter the word to be replaced.

<u>Step4</u>: If the word exists in the document words will be replaced.

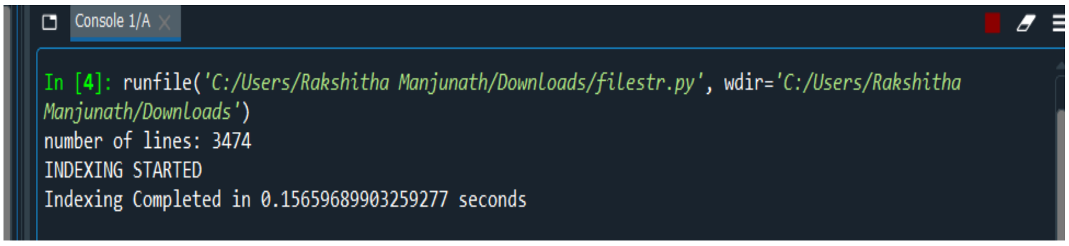
<u>Step5</u>: If no, it terminates and the file is closed.

In the above algorithm it takes the user input whether the word to be modified or not. If the searched word is present in the document and if the user enters ‘yes’ it askes which word to be modified and askes which word to be replaced with. If the user enters ‘no’ it will terminate.

4. Result and Analysis

Snapshots

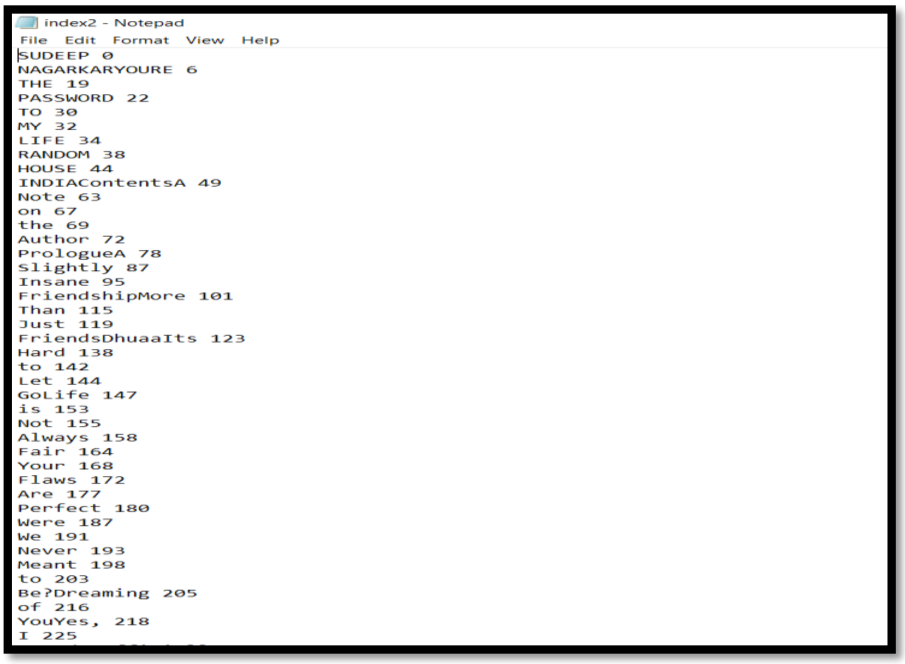
<u>Indexing</u>



```
In [4]: runfile('C:/Users/Rakshitha Manjunath/Downloads/filestr.py', wdir='C:/Users/Rakshitha
Manjunath/Downloads')
number of lines: 3474
INDEXING STARTED
Indexing Completed in 0.15659689903259277 seconds
```

fig 4.1

The above fig firstly returns the total number of lines in the input document. Starts indexing and stores the output file and calculates the time taken to complete indexing operation

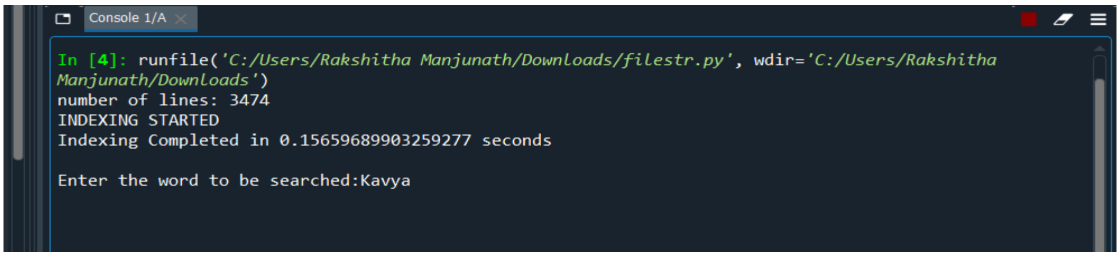


```
SUDEEP 0
NAGARKARYOURE 6
THE 19
PASSWORD 22
TO 30
MY 32
LIFE 34
RANDOM 38
HOUSE 44
INDIAContentsA 49
Note 63
on 67
the 69
Author 72
PrologueA 78
slightly 87
Insane 95
FriendshipMore 101
Than 115
Just 119
FriendsDhuaaIts 123
Hard 138
to 142
Let 144
GoLife 147
is 153
Not 155
Always 158
Fair 164
Your 168
Flaws 172
Are 177
Perfect 180
Were 187
We 191
Never 193
Meant 198
to 203
Be?Dreaming 205
of 216
YouYes, 218
I 225
```

fig 4.2

The above fig represents the output file with the indexes of each word.

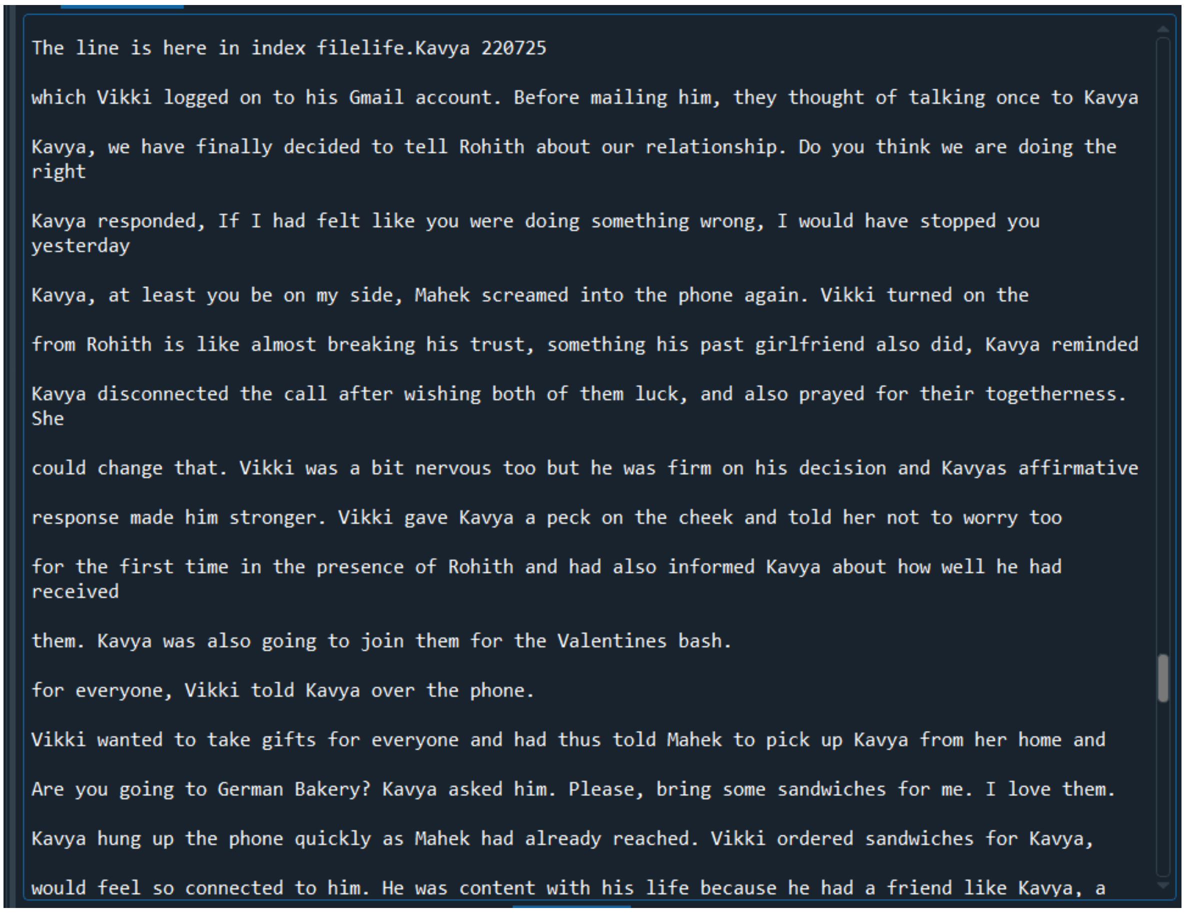
Searching



```
In [4]: runfile('C:/Users/Rakshitha Manjunath/Downloads/filestr.py', wdir='C:/Users/Rakshitha
Manjunath/Downloads')
number of lines: 3474
INDEXING STARTED
Indexing Completed in 0.15659689903259277 seconds

Enter the word to be searched:Kavya
```

fig 4.3

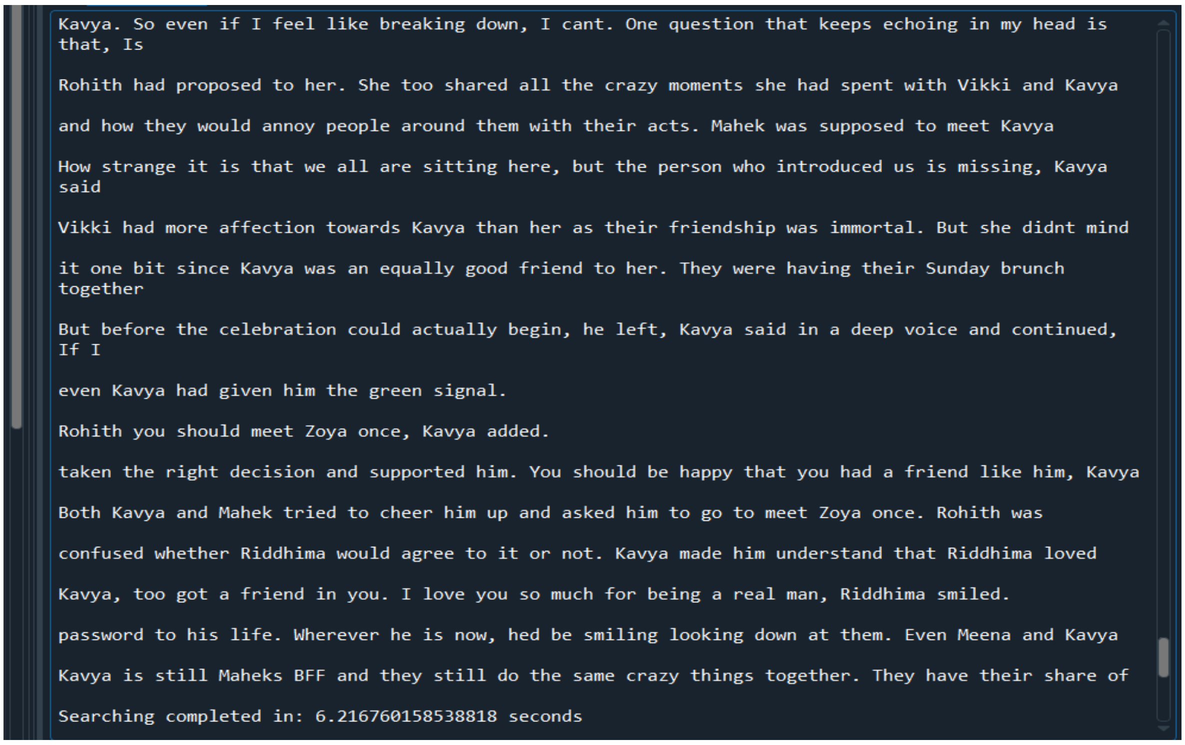
The above fig represents the searching operation where it takes the input from the user, which word to be searched



```
The line is here in index filelife.Kavya 220725

which Vikki logged on to his Gmail account. Before mailing him, they thought of talking once to Kavya

Kavya, we have finally decided to tell Rohith about our relationship. Do you think we are doing the
right

Kavya responded, If I had felt like you were doing something wrong, I would have stopped you
yesterday

Kavya, at least you be on my side, Mahek screamed into the phone again. Vikki turned on the

from Rohith is like almost breaking his trust, something his past girlfriend also did, Kavya reminded

Kavya disconnected the call after wishing both of them luck, and also prayed for their togetherness.
She

could change that. Vikki was a bit nervous too but he was firm on his decision and Kavyas affirmative

response made him stronger. Vikki gave Kavya a peck on the cheek and told her not to worry too

for the first time in the presence of Rohith and had also informed Kavya about how well he had
received

them. Kavya was also going to join them for the Valentines bash.

for everyone, Vikki told Kavya over the phone.

Vikki wanted to take gifts for everyone and had thus told Mahek to pick up Kavya from her home and

Are you going to German Bakery? Kavya asked him. Please, bring some sandwiches for me. I love them.

Kavya hung up the phone quickly as Mahek had already reached. Vikki ordered sandwiches for Kavya,

would feel so connected to him. He was content with his life because he had a friend like Kavya, a
```

fig 4.4

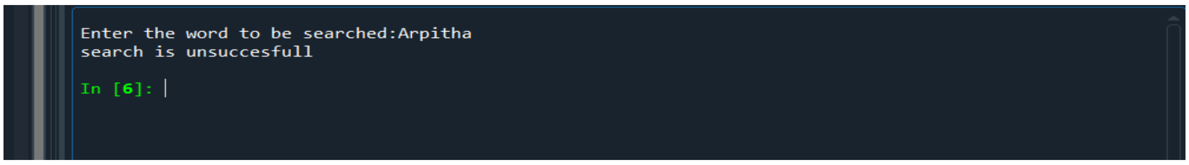
The above fig is the output where it displays indexes and the searched word exists.



```
Kavya. So even if I feel like breaking down, I cant. One question that keeps echoing in my head is
that, Is

Rohith had proposed to her. She too shared all the crazy moments she had spent with Vikki and Kavya

and how they would annoy people around them with their acts. Mahek was supposed to meet Kavya

How strange it is that we all are sitting here, but the person who introduced us is missing, Kavya
said

Vikki had more affection towards Kavya than her as their friendship was immortal. But she didnt mind

it one bit since Kavya was an equally good friend to her. They were having their Sunday brunch
together

But before the celebration could actually begin, he left, Kavya said in a deep voice and continued,
If I

even Kavya had given him the green signal.

Rohith you should meet Zoya once, Kavya added.

taken the right decision and supported him. You should be happy that you had a friend like him, Kavya

Both Kavya and Mahek tried to cheer him up and asked him to go to meet Zoya once. Rohith was

confused whether Riddhima would agree to it or not. Kavya made him understand that Riddhima loved

Kavya, too got a friend in you. I love you so much for being a real man, Riddhima smiled.

password to his life. Wherever he is now, hed be smiling looking down at them. Even Meena and Kavya

Kavya is still Maheks BFF and they still do the same crazy things together. They have their share of

Searching completed in: 6.216760158538818 seconds
```

fig 4.5

The above fig returns the time required to perform search operation.

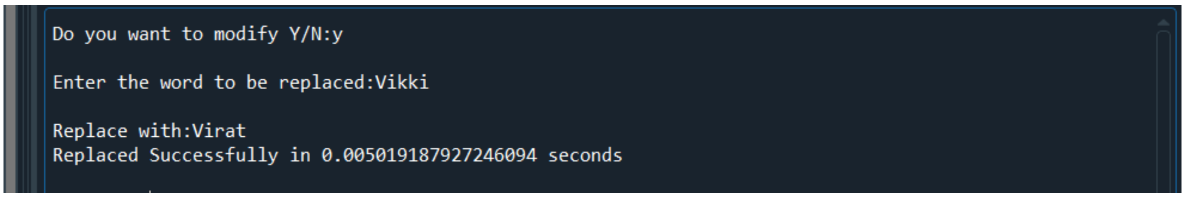


```
Enter the word to be searched:Arpitha
search is unsuccesfull

In [6]: |
```

fig 4.6

When the search word doesn’t exist in the given file.

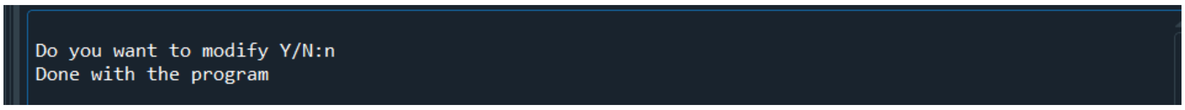
Modifying



```
Do you want to modify Y/N:y

Enter the word to be replaced:Vikki

Replace with:Virat
Replaced Successfully in 0.005019187927246094 seconds
```

fig 4.7

The above fig represents modify operation where the user can replace any word in the file and it returns the time required to perform replace operation

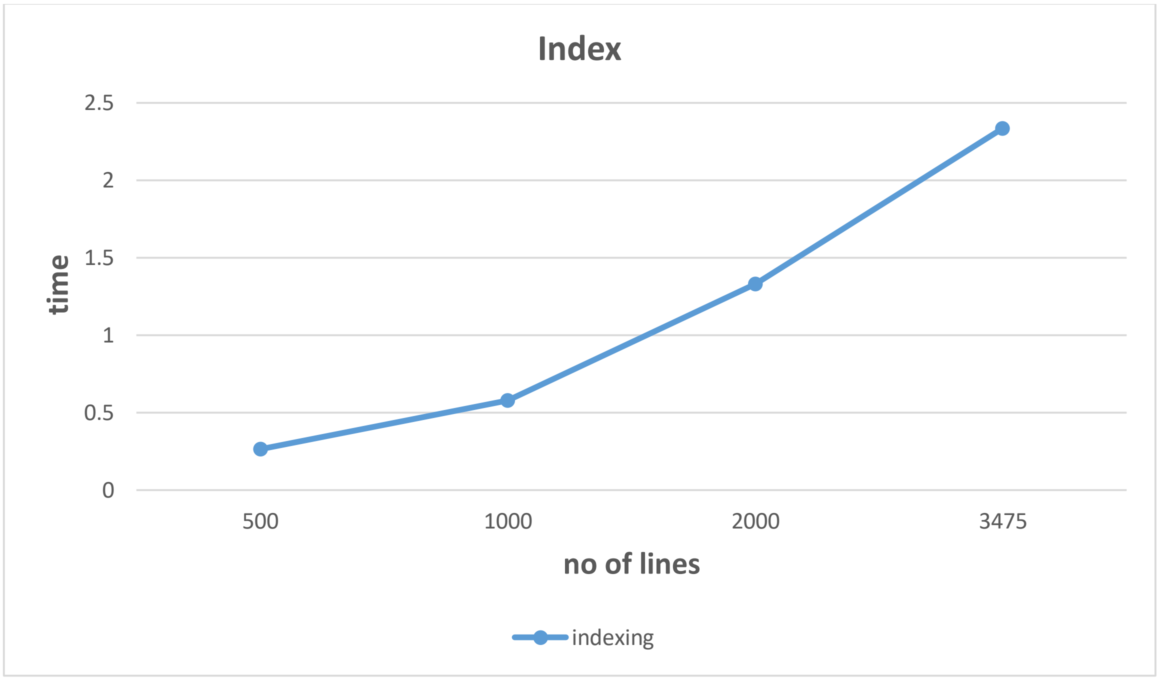


```
Do you want to modify Y/N:n
Done with the program
```

fig 4.8

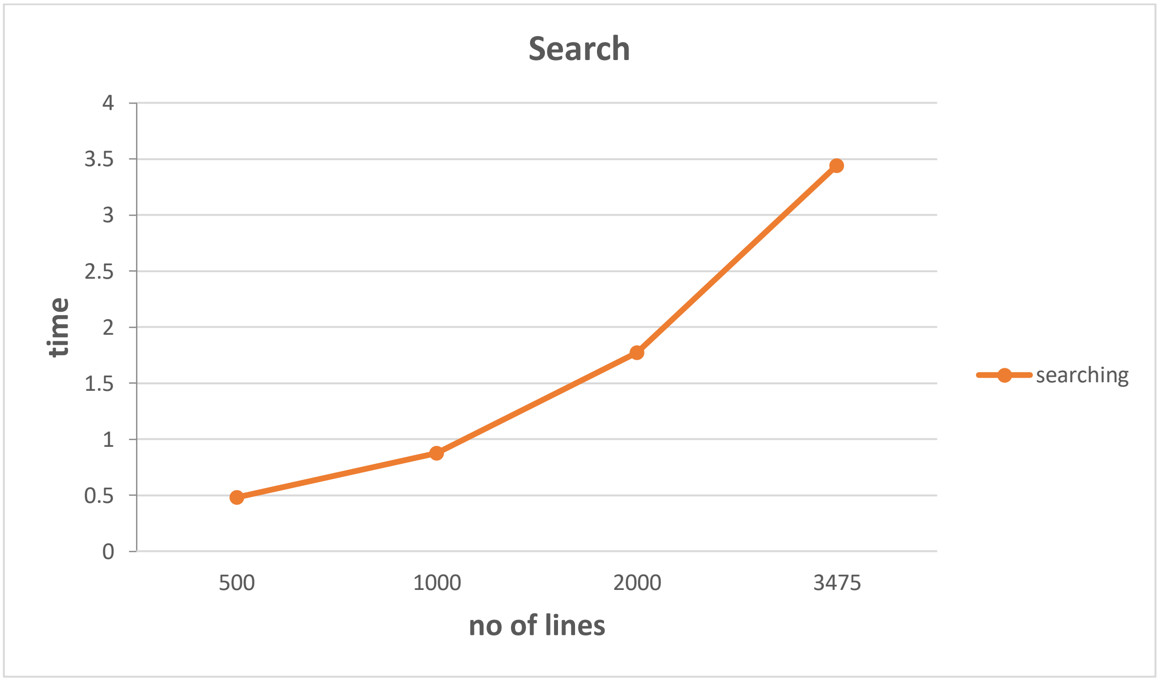
When the search does not exist in the given input document

Analysis



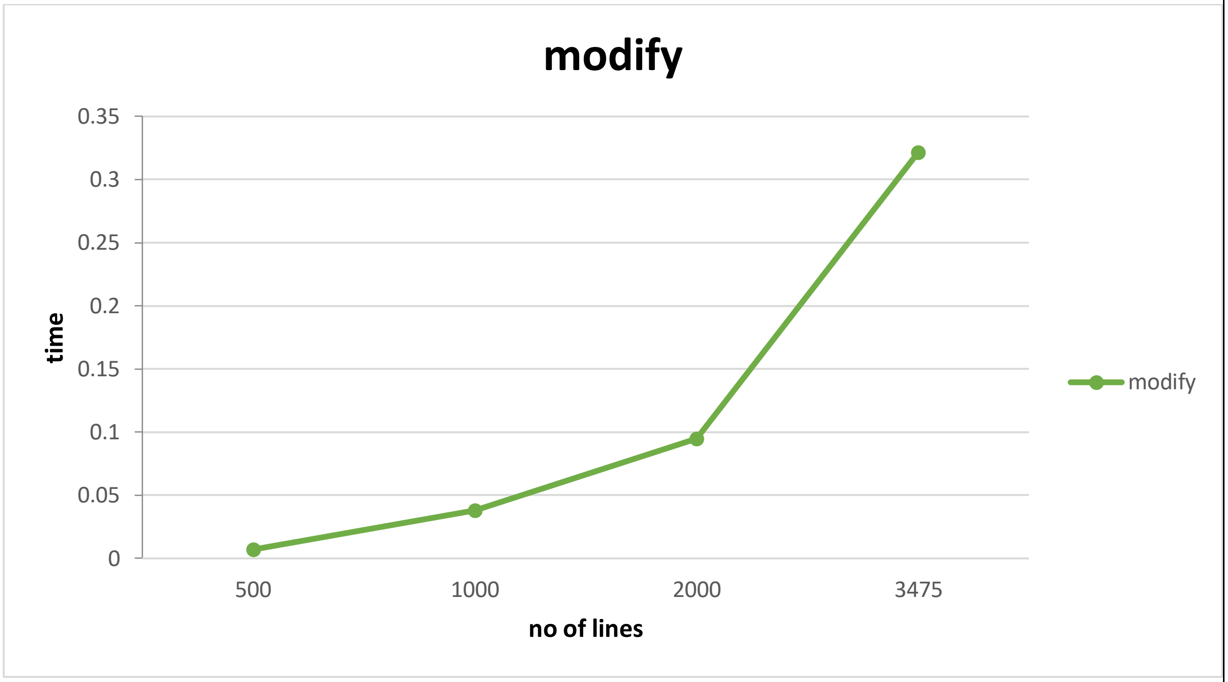
Time taken for indexing

No of lines	**Time taken for indexing(sec)**
500	0.2636046
1000	0.578413
2000	1.32990017
3475	2.33436301



Time taken for searching

No of lines	Time taken for searching(sec)
500	0.479444
1000	0.87526
2000	1.7752
3475	3.443152



Time taken for modifying

No of lines	Time taken for modifying(sec)
500	0.007168
1000	0.03788
2000	0.094628
3475	0.3217594

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 represented in the Y-axis and **No. of lines** in the X-axis. As we increase the input, time drastically increases. Almost exponentially.

5. Future Work and Conclusion

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.

Conclusion

It was a huge learning experience as I took each aspect of the project seriously. My learning curve and also in coding has increased drastically with this mini project. Python has such vast domain features and anything can be accomplished. It was a very good learning experience in terms of time management and learning objectives. I hereby conclude this mini project “Document Processing Using Keywords” successfully with the trust of my senses and to best of my ability.

6. References

[1] **https://www.tutorialspoint.com/python/string_split.htm**

[2] **https://www.geeksforgeeks.org/python-strings/**

[3] **https://app.diagrams.net/**

[4]https://stackoverflow.com/questions/11696472/seek-function

[5] File Structures-An Object Oriented Approach with C++ Michael J. Folk, Bill Zoellick, Greg Riccardi

[6] File Structures Using C++ K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj