# E-BOOK ANALYSIS AND REPRESENTATION

**by**

**RANA GÜL**

**Github - github.com/ranagull**

**Linkedin - linkedin.com/in/ranagül**

**January 2020**

**İZMİR**

# Contents

# 1 INTRODUCTION

I prepared this project as a python project for the Introduction to Computer Engineering course. This project is a python program that finds word sequence. The program takes 1 or 2 book titles from the user and pulls their printable versions from www.wikibooks.com. Then it converts from html to text and saves it to a separate text file on the computer. After deleting the stop words and punctuation marks, it creates a dictionary and does them with the help of a separate txt file. This program that I wrote prints the most words in the book, as much as the number of words taken from the user, if the user has entered 1 book. If the user has entered 2 books, he will first print the common words in descending order and then the distinct words on the screen as many as the number he received from the user. The user also has the right not to choose the number of words. In this case, words are displayed with a default value of 20.

# 2 METHODOLOGY

In this section, I will talk about the methods I use and the right and wrong ways I have done to overcome the problems. I will also explain how I made my code better and used it effectively.

## 2.1 Structure of My Project

First of all, my project consists of a main section and partial function sections. The program that starts in the main section continues by calling functions and prints the output to the last screen. To make my work easier and my code not to be too long, I tried to put it into functions as much as I could and supported it using libraries.

Request library allows me to request information from Http. Beautiful Soap library allows me to easily process the Html file and split the codes to get the information I want. The operator library allows me to edit my dictionary without much effort.

When the program is first run, a question appears on the screen and the user chooses how many books user wants to enter and the code is divided into two accordingly. As a chapter with a book and a chapter with two books.

In the section where a book is processed, we take the name of the book and the number of words the user wants to see and send it to the function named 'Create_Books'.

The Create_Books Function goes to the printable version of the book on the website www.wikibooks.com, merges the name of the book with the url and uses the 'request' library. At this point, I had to extract the formula for the URL of the page, and I saw that it changes in some cases, for example, when the 'status code' of the page is 200 it says / Printable_version and when it is 404 it says / printable_version. After checking this situation too, I use the 'Beautiful Soup' library to extract the content of the page as html. Later, I open a txt file named 'First_book.txt' and write "class": "mw-parser-output" part to this file, which is a specific location in the content section I obtained. Then I open another txt file named 'First_book_words.txt' and print the content I got from the web page by adding a word-shaped line to this file. And then again I put the words in the same file into a separate list with the readlines() command. In all this, my goal is to get rid of extra spaces and get a clean word list. And finally for this list, I call the 'Delete_Punctuation' function and then the 'Delete_Words' function.

In the Delete_Punctuations and Delete_Words functions, I delete the punctuation marks in the form of strings and lists, if they exist in the words in the list, and replace them with spaces. And I divided it by a space and added it to a new list to create a cleaner list of stop words and punctuation.

I then post this list to the 'Create_Dictionary' function. The Create_Dictionary function creates a dictionary from this list and allows me to find out how many of each word.

And after all, I print the most common words in the book for 1 book. For this, I got help from the 'sequential' structure and the 'operator' library. I have listed this dictionary with Sorted. I ordered the words according to their numbers with the operator.itemgetter (1) structure and printed it on the screen by reversing it.

In the section where the two books were selected, I proceeded in the same way until the dictionary creation part I applied to one book. In this section, I opened separate txt files for the second book. Later, I looked through the dictionaries of these two books and found common words and printed them on the screen. Deleting these common words from each of them, I found the distinct words and printed them on the screen.

I used the format method to get a table view while printing on the screen.

## 2.2 Encountered Problems and Solutions

First of all, since I have never pulled data from the internet before, I did research on this subject. I found out which libraries I should use and downloaded them. At this point I struggled to use libraries and got help from some websites [10]. When I first started writing the project, I realized that the texts were missing from the 'p' section, and I converted it to 'body'. But in this case too many words came in than they should have been. Finally, I got the closest result by pulling ("div", {"class": "mw-parser-output"}) from this section.

I had to clear the stop words of the words I got from the book. I had written a separate function for these and I was deleting it from the array, but I was constantly getting blank and compound words and meaningless numbers. I tried to progress by watching a few YouTube videos here [1][2][3]. As a solution to these, I opened a separate function and here I found and deleted the symbols in the words using string punctuation marks. But I still could not solve the word with gaps. I found the remedy to split the word again each time I delete. At the end of all this, I can now get the right words.

I noticed that some words count different from the homework document. The results of distinct words count different from the homework document, but I could not find a way to prevent this from happening because I take a large area of the book to contain words, which leads to an increase in the number of common words, and I also think this is due to the difference in our technique of removing stop words and punctuation. Here I first made a deletion on a list and added unnecessary words and letters to this list. Later, I deleted the unnecessary numbers and symbols in the words through a string variable because I think these are also meaningless. For this reason, I get no

punctuation and no symbol words as output. This situation causes me to get a different output from the homework document. But I tried to achieve maximum similarity by working on my code.

I noticed that some computers gave encoding errors while reading, although I did not get an error on my own computer. I added the command (encoding = 'utf-8'), thanks to my research[7], so that the program can run on every computer that is tried. In this way, I think the code can run smoothly.

Since I couldn't get a regular view while printing on the screen, I did a lot of research [5] and realized that the easiest method is the format() method like print('{:>2d} {:<12s}{:>5d}'.format(TheNumOfPrint, key, count))

I got too many errors about URL. I found that the URLs of some books were different and I could not link this to a formula. As a result of my research, I realized that every web page has a status code [8], and I found that most book pages are 200 but some are 404. In this case, I added if to get a correct result, and if it is 404, I made changes to the last part of the URL.

### 2.3 Improvements

I tried to use functions as much as possible to improve my project. First, I shortened everything by writing it uniformly and then typing them into functions. Thus, a more understandable and effective code has emerged.

I have added to my code in order not to get an error. I added (encoding = 'utf-8') section to avoid getting encode error.

To avoid getting URL errors, I extracted the formula for the link and supported it with if.

I used the format method to make the screen printout look like the homework document and I obtained the table view.

I kept the contents of each book in separate text files and again printed the words of each book in separate text files.

I have used 3 different libraries which are Operator, Beautiful Soap and Request libraries so that my code can run faster and easier. Beautiful Soap and Request allow me to easily extract data from the internet. Operator, on the other hand, helps me easily sort the data I keep in the dictionary.

## 3 EXPERIMENTATION

I've gone through a lot of checks to make sure my code is working correctly. I debugged, tried different ways. I tried to find the best version. I selected different books from the 'wikibooks' and run them in the program to make sure they work with each book. I have tried a lot of Single and Two books and now I'm sure my program is running smoothly.

I've tried to make sure it works across different applications as well. These programs are PyCharm, Vs Code, and IDLE. I also tried to confirm that it works from the command prompt.

## 4 CONCLUSION

Before I started this project, I didn't know much about data processing from the internet. I got information by researching, watching videos, and consulting those who know. And I combined all of these in this project by adding my own knowledge. There were parts where I got a lot of mistakes, but I think I managed them by spending time thinking about them. As a result of all this, the program I wrote can take the book over the web with the name of the book and the number of words it receives from the user and count its meaningful words. If the number of this book is 2, it can compare these words and distinguish between common and different. And it can print all these on the screen as a table, as many times as the user demands.

## APPENDIX A: CODE

My homework code is without comment lines:

```
import requests
from bs4 import BeautifulSoup
import operator
allwords = []
Count_of_book=int(input("Please enter the number of book: "))
def Create_Books(bookname,x):
    Book_Name_link = bookname.replace(" ", "_")
    Book_Name_link = Book_Name_link.replace("'", "%27")
    All_of_words = []
    if x == 1:
        Url = "https://en.wikibooks.org/wiki/"
        Printable_Version = "/Print_version"
        r = requests.get(Url + Book_Name_link + Printable_Version)
        if r.status_code == 404:
            Printable_Version = "/print_version"
            r = requests.get(Url + Book_Name_link +
Printable_Version)
        soup = BeautifulSoup(r.content, "html.parser")

        Book_text = open("First_book.txt", "w",encoding='utf-8')
        Book_text1 = open("First_book_words.txt", "w",
encoding='utf-8')
        for word in soup.find_all("div",{"class":"mw-parser-
output"}):
            content = word.text
            Book_text.write(content)
            Full_words = content.lower().split()
            for w in Full_words:
                Book_text1.write(w+"\n")
        Book_text.close()
        Book_text1.close()

        Book_text11 = open("First_book_words.txt", "r",
encoding='utf-8')
        All_of_words = Book_text11.readlines()
        Book_text11.close()

        All_of_words = Delete_Punctuation(All_of_words)
        All_of_words = Delete_Words(All_of_words)
    elif x == 2:
        Url = "https://en.wikibooks.org/wiki/"
        Printable_Version = "/Print_version"
        r = requests.get(Url + Book_Name_link + Printable_Version)
        if r.status_code == 404:
            Printable_Version = "/print_version"
            r = requests.get(Url + Book_Name_link +
Printable_Version)
        soup = BeautifulSoup(r.content, "html.parser")

        Book_text = open("Second_book.txt", "w",encoding='utf-8')
```

```
        Book_text1 = open("Second_book_words.txt",
"w",encoding='utf-8')
        for word in soup.find_all("div",{"class":"mw-parser-
output"}):
            content = word.text
            Book_text.write(content)
            Full_words = content.lower().split()
            for w in Full_words:
                Book_text1.write(w+"\n")
        Book_text.close()
        Book_text1.close()

        Book_text11 = open("Second_book_words.txt", "r",
encoding='utf-8')
        All_of_words = Book_text11.readlines()
        Book_text11.close()

        All_of_words = Delete_Punctuation(All_of_words)
        All_of_words = Delete_Words(All_of_words)
    return  All_of_words
def Delete_Words (allwords):
    Filtered_Words=[]
    Stop_Words= ["etc", "us", "a", "b", "c", "d", "e", "f", "g",
"h", "i", "j", "k",
                 "l", "m", "n", "o", "p", "r",'s', "u", "v", "y",
"z", "w", "x", "q", "I", "me",
                 "my", "myself", "we", "our", "ours", "ourselves",
"you", "your", "yours",
                 "yourself", "yourselves", "he", "him", "his",
"himself", "she", "her", "hers",
                 "herself", "it", "its", "itself", "they", "them",
"their", "theirs", "themselves",
                 "what", "which", "who", "whom", "this", "that",
"these", "those", "am", "is", "are",
                 "was", "were", "be", "been", "being", "have",
"has", "had", "having", "do", "does", "did",
                 "doing", "a", "an", "the", "and", "but", "if",
"or", "because", "as", "until", "while", "of",
                 "at", "by", "for", "with", "about",
"against","name","value","next","first", "between", "into",
"through", "during", "before",
                 "after", "above", "below", "to", "from",
"up","line", "down", "in", "out", "on", "off", "over", "under",
                 "again", "further", "then","ing", "once", "here",
"there", "when", "where", "why", "how", "all", "any",
                 "both", "each", "few", "more", "most",
"other","use", "some", "such", "no", "nor", "not", "only", "own",
                 "same", "so", "than", "too", "very", "s", "t",
"can", "will", "just", "don", "should", "now","'m", "'s", "'t",
"'ll", "'ve", "'re",
                 '—', '=', '==', '\t', '/', '0', '1', '2', '3', '4',
'5', '6', '7', '8', '9', '.' , ',' , '!' ,'?','(',')' ,';',
                 '[', ']', '\n' , '"' , ':' , '-', '  ' , '~' , '@'
, '^' , '#' , '%' , '$' , '&' , '*' , '_' ,'`' , '{' , '}' , '|',
'>' , '<','←','↑','→']
```

```
    words = []
    for word in allwords:
        for stops in Stop_Words:
            if stops == word:
                word = word.replace(stops," ").strip()
        words =  word.split()
        if (len(word) > 0):
            Filtered_Words += words
    return Filtered_Words
def Delete_Punctuation(allwords):
    Filtered_Words = []
    words = []
    Punctuation ='!"#$%&()*, -./:;<=>?@[\]^_`{|}~0123456789'+"'"
    for word in allwords:
        for punc in Punctuation:
            if punc in word:
                word = word.replace(punc," ").strip()
        words =  word.split()
        if (len(word) > 0):
            Filtered_Words += words
    return Filtered_Words
def Create_Dictionary(allwords):
    Count_of_word= {}
    for word in allwords:
        if len(word) != 0 :
          if word in Count_of_word:
                word= word.replace(" ","")
                Count_of_word[word] += 1
          else:
                word= word.replace(" ","")
                Count_of_word[word] = 1
    return Count_of_word
if Count_of_book == 1:
    First_Book_Name = input("Please Enter the Book Name : ")
    The_choose = input("Would You Like to Choose Word Frequencies
Count? Please enter Y or N ! ")
    if The_choose == "Y":
        Count_of_print_word = int(input("How Many Word Frequencies
You Wish To See : "))
    else:
        Count_of_print_word = 20
    print()
    print("BOOK 1 : "+ First_Book_Name)
    print("NO WORD          FREQ_1")

    Allwords = Create_Books(First_Book_Name,1)
    final_words = Create_Dictionary(Allwords)

    Total_word_count = 0
    Count_of_Finalwords = len(final_words)
    TheNumOfPrint = 0
    for key, count in sorted(final_words.items() ,reverse=True, key
=operator.itemgetter(1)):
        Total_word_count += count
        TheNumOfPrint = TheNumOfPrint + 1
```

```
        if TheNumOfPrint <= Count_of_Finalwords -
Count_of_print_word:
            print('{:>2d}
{:<12s}{:>5d}'.format(TheNumOfPrint,key,count))
            if (TheNumOfPrint == Count_of_print_word):
                break
elif Count_of_book == 2:
    First_Book_Name = input("Please Enter the Book Name : ")
    Second_Book_Name = input("Please Enter the Book Name : ")
    The_choose = input("Would You Like to Choose Word Frequencies
Count? Please enter Y or N ! ")
    if The_choose == "Y":
        Count_of_print_word = int(input("How Many Word Frequencies
You Wish To See : "))
    else:
        Count_of_print_word = 20

    print()
    print("BOOK 1 : "+ First_Book_Name)
    print("BOOK 2 : " + Second_Book_Name)
    print("COMMON WORDS")
    print("NO WORD           FREQ_1 FREQ_2 FREQ_SUM")

    FirstBook_Allwords =  Create_Books(First_Book_Name,1)
    SecondBook_Allwords = Create_Books(Second_Book_Name,2)

    final_words_first = Create_Dictionary(FirstBook_Allwords)
    final_words_second = Create_Dictionary(SecondBook_Allwords)

    Mostly_Common_words = {}
    for key1, count1 in sorted( final_words_first.items(),
reverse=True, key=operator.itemgetter(1)):
        for key2 , count2 in sorted(final_words_second.items(),
reverse=True, key=operator.itemgetter(1)):
            if key1 == key2:
                if key1 in Mostly_Common_words:
                    Mostly_Common_words[key1] += (count1 + count2)
                else:
                    Mostly_Common_words[key1] = (count1 + count2)

    Count_of_Finalwords = len(Mostly_Common_words)
    TheNumOfPrint = 0
    Total_word_count = 0
    str = " "
    for key1,count1 in  sorted( Mostly_Common_words.items(),
reverse=True, key=operator.itemgetter(1)):
        TheNumOfPrint = TheNumOfPrint + 1
        Total_word_count += count1
        if TheNumOfPrint <= Count_of_Finalwords -
Count_of_print_word:
            print('{:>2d} {:<12s}  {:>5d}  {:>5d}
{:>5d}'.format(TheNumOfPrint,key1,final_words_first[key1],final_word
s_second[key1],count1))
            if(TheNumOfPrint == Count_of_print_word):
                break
```

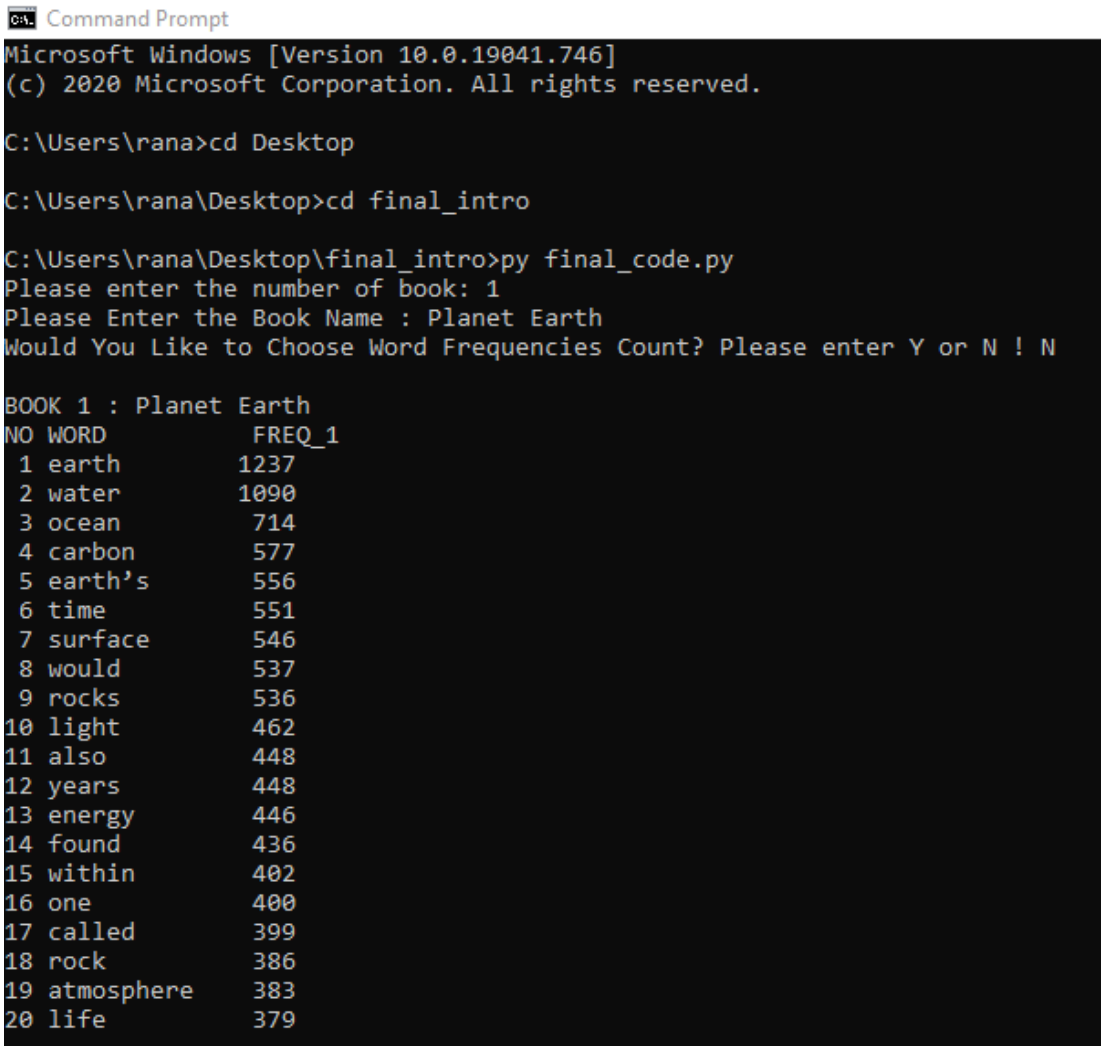
```
    print()
    print("BOOK 1 : "+ First_Book_Name)
    print("DISTINCT WORDS")
    print("NO WORD           FREQ_1")

    for key1, count1 in sorted( final_words_first.items(),
reverse=True, key=operator.itemgetter(1)):
        for key2 , count2 in sorted(final_words_second.items(),
reverse=True, key=operator.itemgetter(1)):
            if key1 == key2:
                if key1 in Mostly_Common_words:
                    if key1 in final_words_first :
                        del final_words_first[key1]
                    if key2 in final_words_second:
                        del final_words_second[key2]
    Total_word_count = 0
    TheNumOfPrint = 0
    for key, count in sorted(final_words_first.items()
,reverse=True, key =operator.itemgetter(1)):
        Total_word_count += count
        TheNumOfPrint = TheNumOfPrint + 1
        if TheNumOfPrint <= Count_of_Finalwords -
Count_of_print_word:
            print('{:>2d}
{:<12s}{:>5d}'.format(TheNumOfPrint,key,count))
            if (TheNumOfPrint == Count_of_print_word):
                break
    print()
    print("BOOK 2 : "+ Second_Book_Name)
    print("DISTINCT WORDS")
    print("NO WORD           FREQ_1")

    Total_word_count = 0
    TheNumOfPrint = 0
    for key, count in sorted(final_words_second.items()
,reverse=True, key =operator.itemgetter(1)):
        Total_word_count += count
        TheNumOfPrint = TheNumOfPrint + 1
        if TheNumOfPrint <= Count_of_Finalwords -
Count_of_print_word:
            print('{:>2d}
{:<12s}{:>5d}'.format(TheNumOfPrint,key,count))
            if (TheNumOfPrint == Count_of_print_word):
                break
```

## APPENDIX B: SCREENSHOTS OF MY USE CASES

```
C:\Users\rana\Desktop\final_intro>py final_code.py
Please enter the number of book: 1
Please Enter the Book Name :  Non-Programmer's Tutorial for Python 2.6
Would You Like to Choose Word Frequencies Count? Please enter Y or N ! Y
How Many Word Frequencies You Wish To See : 25

BOOK 1 :  Non-Programmer's Tutorial for Python 2.6
NO WORD          FREQ_1
 1 print         521
 2 number        278
 3 program       179
 4 python        156
 5 +             153
 6 input         147
 7 list          142
 8 function      131
 9 numbers       101
10 menu           99
11 item           98
12 true           96
13 type           92
14 string         92
15 text           91
16 run            82
17 license        81
18 file           76
19 document       75
20 demolist       73
21 index          68
22 return         68
23 false          67
24 raw            66
25 choice         66
```

```
C:\Users\rana\Desktop\final_intro>py final_code.py
Please enter the number of book: 2
Please Enter the Book Name : Non-Programmer's Tutorial for Python 2.6
Please Enter the Book Name : Non-Programmer's Tutorial for Python 3
Would You Like to Choose Word Frequencies Count? Please enter Y or N ! N

BOOK 1 : Non-Programmer's Tutorial for Python 2.6
BOOK 2 : Non-Programmer's Tutorial for Python 3
COMMON WORDS
NO WORD            FREQ_1 FREQ_2 FREQ_SUM
 1 print            521    545    1066
 2 number           278    298     576
 3 python           156    214     370
 4 program          179    177     356
 5 list             142    160     302
 6 +                153    141     294
 7 input            147    138     285
 8 function         131    123     254
 9 numbers          101    140     241
10 menu              99    109     208
11 true              96     99     195
12 item              98     96     194
13 type              92     95     187
14 string            92     88     180
15 file              76     98     174
16 text              91     74     165
17 run               82     78     160
18 demolist          73     69     142
19 choice            66     76     142
20 false             67     69     136
```

```
BOOK 1 : Non-Programmer's Tutorial for Python 2.6
DISTINCT WORDS
NO WORD            FREQ_1
 1 raw             66
 2 sections        31
 3 title           27
 4 invariant       23
 5 texts           20
 6 entitled        15
 7 preserve        13
 8 publisher       12
 9 transparent     11
10 gnu              9
11 published        9
12 provided         9
13 mmc              9
14 titles           8
15 history          8

BOOK 2 : Non-Programmer's Tutorial for Python 3
DISTINCT WORDS
NO WORD            FREQ_1
 1 path            11
 2 environment      8
 3 pip              7
 4 wt               6
 5 arithmetic       5
 6 rt               5
 7 closing          4
 8 libraries        4
 9 bigger           4
10 https            3
11 nix              3
12 ending           3
13 tgz              3
14 panel            3
15 prog             3
```


Command Prompt
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.
C:\Users\rana>cd Desktop
C:\Users\rana\Desktop>cd final_intro
C:\Users\rana\Desktop\final_intro>py final_code.py
Please enter the number of book: 1
Please Enter the Book Name : Planet Earth
Would You Like to Choose Word Frequencies Count? Please enter Y or N ! N
BOOK 1 : Planet Earth
NO WORD FREQ_1
1 earth 1237
2 water 1090
3 ocean 714
4 carbon 577
5 earth's 556
6 time 551
7 surface 546
8 would 537
9 rocks 536
10 light 462
11 also 448
12 years 448
13 energy 446
14 found 436
15 within 402
16 one 400
17 called 399
18 rock 386
19 atmosphere 383
20 life 379

**REFERENCES**


[1] Mustafa Murat Coşkun/Yazılım Bilimi.Python3 Dersleri 43-Bir Web Sitesindeki Kelime Frekansı 1 (İnternetten Veri Çekmek) (27 June 2016). Accessed: January 10, 2021. [Online Video]. Avaible: https://www.youtube.com/watch?v=II9mlqPmgII&ab_channel=Yaz%C4%B1l%C4%B1mBilimi

[2] Mustafa Murat Coşkun/Yazılım Bilimi.Python3 Dersleri 43-Bir Web Sitesindeki Kelime Frekansı 2 (İnternetten Veri Çekmek) (28 June 2016). Accessed: January 10, 2021. [Online Video]. Avaible: https://www.youtube.com/watch?v=4noJ2hJuKx4&list=PLIHume2cwmHehcxQE1XZieL21syR3m3tR&index=44&t=178s&ab_channel=Yaz%C4%B1l%C4%B1mBilimi

[3] Mustafa Murat Coşkun/Yazılım Bilimi.Python3 Dersleri 44-Bir Web Sitesindeki Kelime Frekansı 3 (İnternetten Veri Çekmek) (28 June 2016). Accessed: January 10, 2021. [Online Video]. Avaible: https://www.youtube.com/watch?v=5RNdmnz-meg&list=PLIHume2cwmHehcxQE1XZieL21syR3m3tR&index=45&t=45s&ab_channel=Yaz%C4%B1l%C4%B1mBilimi

[4] January 11 , 2021.[Online]. Avaible: https://www.programiz.com/python-programming/dictionary

[5] January 15 , 2021.[Online]. Avaible: https://scientificallysound.org/2016/10/17/python-print3/

[6] January 17 , 2021.[Online]. Avaible: http://hilite.me/

[7] January 14 , 2021.[Online]. Avaible: https://stackoverflow.com/questions/14242486/python-why-am-i-getting-a-unicodedecodeerror/14242541#14242541

[8] January 14 , 2021.[Online]. Avaible: https://www.sinanerdinc.com/python-click-modulu

[9] January 17, 2021. [Online]. Avaible: https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf

[10] January 9, 2021. [Online]. Avaible: https://www.mygreatlearning.com/blog/open-source-python-libraries/