DOKUZ EYLÜL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

E-BOOK ANALYSIS AND REPRESENTATION

Assignment Report

by

Deniz Küçükkara

January 2020

İZMİR

Contents

1 INTRODUCTION

This project aims to find the book or books the user wants from “Wikisource” and show the most repetitive words as many times as the user wants. In addition, if the user is comparing two books, "distinct words" are shown in the same way. If the user doesn't enter the numbers of the numbers she/he wants to see, the default 20 words are shown. After getting the data from the user, the code checks for any errors thanks to the "try-except" commands. If the user has entered the wrong input, it asks the user if they want to try again. If the user wants to change and rewrite the inputs, he can rewrite it thanks to(while) loop. Otherwise, if he wants to exit, he presses the "E" key and leaves the loop. Finally, it performs data extraction and comparison, which I will talk about in more detail below. To conclude, the program continues until the user enters the correct data or wants to exit, and then takes the necessary actions, which I will explain in detail in the following sections, according to the user's request.

2 METHODOLOGY

In this project, I used the "while" loop, "try-except" command, and the "for" loop extensively. I learned the dictionary structure and used them extensively, and I switched between dictionaries and lists to check the number of words. Besides, I learned to use dic.values (), dic.keys () and dic.items () patterns correctly. I did not know that int values cannot be returned in the "for" loop in python, after realizing this I started to use the "while" command where I needed to return a number. But the part I got with the most error was "bound of arrays". After I counter this problem many times, I started to do practice in the lists and finally learned to use the append command more effectively and took care of this problem.

2.1 Structure of Your Project

First, I downloaded the software called pip3 to download the necessary modules(requests, beautifulsoup). I want to explain my code in the order I wrote after here:

Firstly, I asked the user if he/she only wanted to see the words of one book or if he/she wanted to compare the words of two books. If the user wants to learn about just one book, they press 1, and if they want to select the other option, they press 2. Accordingly, the code was entering an "if" block. There were long codes here that I would write inside the blocks over and over. I used functions to avoid this. If the user presses 1, I made him ask how many of the most repetitive words he wants to see. While doing this, I used "try-except" because if the user does not enter anything, the default value of 20 should be selected and the code was broken because the value of the input is an integer. I made the number entered in the "Except" command equal to 20, so the default setting was used if the user does not enter anything. After that, I correctly added the name of the book entered to the URL part of the site and defined the extraction_data () function that I will use to extract data. I added /Print_version of the name entered when this function arrives and

I used the "requests module" required to reach the site. Then I checked whether it was reached with the command "status_code = 200 because when the page is written incorrectly, the status code value becomes smaller or bigger than 200. If the page cannot be reached, I tried the same steps as /print_version. I opened the file and used the beautifulsoup module, which is the second module I used for the project. if it's time to process the first book, I opened the .txt file which name is first_book, the same way when code process the second book it is open .txt file which name is second_book. Thus, when users choose option two, I printed not only one text file but 2 separate text files and obtained a better code. "After these operations, I quit the function and I open try except for If .txt files are empty or incorrect if it is still wrong I gave this output (the book you are looking for is not available on Wikisource) I gave a quick error so the code stopped without crushing. Then I wanted to remove the "stop words" and "punctuation marks" in this file and I opened another function for it. In this function, firstly the text file containing the book is opened and separated according to the spaces and transferred to a list. I used a "for" loop to check each word one by one after

transferring it. In order to avoid confusion, I first made all the letters lowercase when every word came. Then I had it checked for punctuation anywhere in the word and if there were punctuation I deleted them and reprinted the word. After the words got rid of the punctuation, I had to have them checked for "stop words". That's why I first opened a list of "stop words". and if the word from the for loop is equal to any word in this list, the word is deleted. I assigned the undeleted words to a new list. This list is made up entirely of elements that are words I can use. After that, I defined a new function to find which words pass in the text and how many times they pass in the text. I opened a dictionary here and started passing each word one by one with the "for" loop. In this loop, I checked whether the dictionary contains the word or not. If it does not contain the word it adds as an element. If the dictionary contains the word its value increases by one. In this way, I easily found how many words occurred in the text.
After that, I switched to a new function. Now I had to sort the words in ascending order. I could not find the correct way to do this in the dictionary template, so I assigned the keys and values to 2 different lists. Then I opened a "for" loop and checked the values in binary. The bigger one was at the top of the list, and when the loop was finished, the list was ready in a sequence. Then I opened a "while" loop that continues as many words as the user wants to see, adjusted the format, and printed the words. This is how things go when the user only wants to see a book. If the user wants to compare 2 books, the same process happens for both books and additionally switches to a function called distinct words. This function has a nested for loop and this loop asks the question: Is the word in the book (a) in the book (b)? If not, the counter increases and at the end of the loop if this counter is equal to the length of the book, it means it does not pass and is printed. Since it is a function, I could only change and use variables for 2 books and I came to the end of my project.

2.2 Encountered Problems and Solutions

The first problem I encountered was (/ Print_Version) at the end of the URL. In some Books, it could be written in lower case. I solved this problem by using "if-else", first try the big version and if the status code is not 200 (I talked about what status code means in more detail above part) it goes into the "else" pattern and tries to print it with a small "p" here. This way, the problem disappears. Another problem I encountered was with the dictionaries. I wanted to compare the elements in the dictionaries, but while changing the elements in them, I could not change them because they became "tuple" or I was saving them over and over. I solved this problem by transferring the dictionary to 2 separate lists. I was able to do my transactions with the lists very easily. Another problem I encountered was related to the format. Although I know the format pattern, I was not fully in control. I referenced from the correct sites and managed to print it correctly. The biggest problem I encountered was the new modules "beautifulsoup" and "requests" that I learned about. I had to research them for a long time from several sources and then practice for hours. But in the end, I fixed all the problems and prepared a code without any problems.

2.3 Improvements

As an improvement, I did my best to keep the code from crashing. If the user enters the wrong book name, the program understands that and, closes after specifying it. If the user wants to compare more than 2 books the code shows a page like this" You cannot select more than 2 or less than 1 book. Press 'E' to exit to 'A' to try again.". If the user press "E" program will close. Another improvement is I took care of the capitalization problems and "_" problems in the URL part. Furthermore, If the user enters a string even though the input part is int, I stated the problem and asked him to try again. Finally, I adjusted the format of the printouts.

3 EXPERIMENTATION

Case 1 :

Input : 1

Book name : Non-Programmer's Tutorial for Python 2.6

Word number : 20

Output :

```
NO   WORD           FREQ_1
 1   print          515
 2   number          246
 3   program        177
 4   +               151
 5   python          143
 6   value            131
 7   list              130
 8   function         130
 9   true             95
 10   text            85
 11   item            84
 12   menu             81
 13   run             81
 14   license         80
 15   string          80
 16   input           74
 17   type            73
 18   use             72
 19   document          70
 20   numbers           67
```

Case 2 :

Input : 2

Book 1 Name : Non-Programmer's Tutorial for Python 3

Book 2 Name : Planet Earth

Word Number : 5

Output :

```
COMMON WORDS
 NO   WORD            FREQ_1        FREQ_2        FREQ_SUM
 1    print            526             8           534
 2    number           253           154           407
 4    program          177            11           188
 5    list             140            10           150
BOOK 1 :  Non-Programmer's_Tutorial_for_Python_3
DISTINCT WORDS
 NO   WORD            FREQ_1
 1    python          186
 2    menu             87
 3    file             75
 4    prev             59
 5    def              58
BOOK 2 :  Planet_Earth
DISTINCT WORDS
 NO   WORD            FREQ_1
 1    earth           1162
 2    ocean             680
 3    earths           558
 4    carbon            557
 5    surface           545
```

4 CONCLUSION

To conclude, I have greatly increased my knowledge of python and improved my practice in this project. I can write much faster than before and now I know 2 new modules. Although this project aims to research and learn, some parts we haven't learned have really challenged me. It would be better if I had a little idea about those places. In this project, I saw how beneficial it is to plan ahead and to stick to a certain plan, and realize the project.

APPENDIX A: CODE

```
from bs4 import BeautifulSoup as bs
import requests

firstbook_list_word_for_repetition = []
secondbook_list_word_for_repetition = []
lastkeylist = []
lastvaluelist = []
keyfor1_stbook = []
valuesfor1_stbook = []
keyfor2_ndbook = []
valuesfor2_ndbook = []
first_book =""
exit = True

second_book =""
manybooks = 0
basic_adress = "https://en.wikibooks.org/wiki/" #basic_adress is the
link required to direct us to the site
stopwords =["a","'", "about", "above", "above", "across", "after",
"afterwards", "again", "against", "all",
            "almost", "alone", "along", "already",
"also","although","always","am","among", "amongst", "amoungst",
            "amount",  "an", "and", "another",
"any","anyhow","anyone","anything","anyway", "anywhere", "are",
"around", "as",  "at",
            "back","be","became", "because","become","becomes",
"becoming", "been", "before", "beforehand", "behind", "being",
"below"
            , "beside", "besides", "between", "beyond", "bill",
"both", "bottom","but", "by", "call", "can", "cannot", "cant", "co",
"con", "could",
            "couldnt", "cry", "de", "describe", "detail", "do",
"done", "down", "due", "during", "each", "eg", "eight", "either",
"eleven","else",
            "elsewhere", "empty", "enough", "etc", "even", "ever",
"every", "everyone", "everything", "everywhere", "except", "few",
"fifteen", "fify", "fill",
            "find", "fire", "first", "five", "for", "former",
"formerly", "forty", "found", "four", "from", "front", "full",
"further", "get", "give", "go", "had",
            "has", "hasnt", "have", "he", "hence", "her", "here",
"hereafter", "hereby", "herein", "hereupon", "hers", "herself",
"him", "himself", "his", "how", "however",
            "hundred", "ie", "if", "in", "inc", "indeed",
"interest", "into", "is", "it", "its", "itself", "keep", "last",
"latter", "latterly", "least", "less", "ltd",
            "made", "many", "may", "me", "meanwhile", "might",
"mill", "mine", "more", "moreover", "most", "mostly", "move",
"much", "must", "my", "myself", "name",
            "namely", "neither", "never", "nevertheless", "next",
"nine", "no", "nobody", "none", "noone", "nor", "not", "nothing",
"now", "nowhere", "of", "off", "often",
            "on", "once", "one", "only", "onto", "or", "other",
"others", "otherwise", "our", "ours", "ourselves", "out", "over",
"own","part", "per", "perhaps", "please",
            "put", "rather", "re", "same", "see", "seem", "seemed",
"seeming", "seems", "serious", "several", "she", "should", "show",
"side", "since", "sincere", "six",
            "sixty", "so", "some", "somehow", "someone",
"something", "sometime", "sometimes", "somewhere", "still", "such",
"system", "take", "ten", "than", "that", "the",
            "their", "them", "themselves", "then", "thence",
"there", "thereafter", "thereby", "therefore", "therein",
"thereupon", "these", "they", "thick", "thin",
            "third", "this", "those", "though", "three", "through",
"throughout", "thru", "thus", "to", "together", "too", "top",
"toward", "towards", "twelve", "twenty",
            "two", "un", "under", "until", "up", "upon", "us",
"very", "via", "was", "we", "well", "were", "what", "whatever",
"when", "whence", "whenever", "where",
            "whereafter", "whereas", "whereby", "wherein",
"whereupon", "wherever", "whether", "which", "while", "whither",
"who", "whoever", "whole", "whom", "whose",
            "why", "will", "with", "within", "without", "would",
"yet", "you", "your", "yours", "yourself", "yourselves", "the",
"won't", "non", "b", "line", "i","$","k"]
punctuation_marks = '!,.;:(#?[]&)=><&@^%_-0123456789*/`{}"|~\n\t'


def max_word_finder_for_dic(txt_file, purified_words) :  #Here the
book saved in .txt opens. Then stop words and punctuation are
deleted.

    book_file = open(txt_file)
    book_readline = book_file.readlines()
    book_file.close()
    for lines in book_readline:
        line = lines.split(" ")

        for words in line:
            words = words.lower() #All letters are shrunk to
distinguish words

            for marks in punctuation_marks:
                if marks in words:
                    index_of_letters = words.index(marks) # if the
word has a sign, it is deleted and a space is put in its place
                    words = words.replace(marks, " ")
                    words = words[0:index_of_letters]

            check_stopwords = 0
            for stop_words in stopwords: #not saved if word is stop
word
                if (words == stop_words) or words == "":
                    check_stopwords = 1
                    break
            if check_stopwords == 0 : # if it is not a stop word
it's append the list
                purified_words.append(words)
    dictionaries_for_count(purified_words)



def dictionaries_for_count(lists) :
    word_list = dict()

    for words in lists :
        if words not in word_list : # if word not in dict it will
saved in there
            word_list[words] = 1
        elif words in word_list : # if dictionary contains the word
words.value getting increase
            word_list[words]+=1


    if s==0: # s = 0 means code checks first book
        from_big_to_small(keyfor1_stbook, valuesfor1_stbook,
word_list)

    elif s!=0 : #s != 0 means code checks second book
        from_big_to_small(keyfor2_ndbook, valuesfor2_ndbook,
word_list)



def from_big_to_small(key_list, value_list, word_list) :

    for values in word_list.values(): #The dictionary is saved in
the list to make the necessary checks more conveniently below.
        value_list.append(values)
    for keys in word_list.keys():
        key_list.append(keys)

    for values in range(len(word_list.keys())): #Sorting from large
to small by performing double checks
```

```
        for values2 in range(len(word_list.keys())):
             if value_list[values] > value_list[values2]:
                value_list[values], value_list[values2] =
value_list[values2],value_list[values]
                key_list[values], key_list[values2] =
key_list[values2], key_list[values]


    d = 0
    while (word_input - 1 >= d): #the user's desired number of big
words are saved in the list
        lastkeylist.append(key_list[d])
        lastvaluelist.append((value_list[d]))
        d += 1


def just_first_book_print(key_list, value_list) :
    d=0
    while (word_input - 1 >= d): #prints for first_book option
        print(' {:<4d} {:<13s} {:<15d}'.format(d + 1, key_list[d],
value_list[d]))
        d += 1
def print_for_double(key_list, value_list) :
    first_book_value_calc = -1

    for i in range(word_input): # print for double option
        first_book_value_calc += 1
        k = -1
        for key2 in key_list:
            k += 1
            if key2 == lastkeylist[first_book_value_calc]:
                print(' {:<4d} {:<13s} {:<15d} {:<16d}{:<15d}
'.format(first_book_value_calc + 1,
lastkeylist[first_book_value_calc],

lastvaluelist[first_book_value_calc], value_list[k],

(value_list[k] + lastvaluelist[first_book_value_calc])))

def distinct(compare_one,compare_two, compare_one_value) :
    l=1
    m=-1
    for key in compare_one : #in these loops it controls distinct
words. If they don't contain same words k become len(list) and code
prints the key
        k=0
        m+=1
        for key2 in compare_two :
            if key != key2 :
                k+=1
        if k== len(compare_two):
            print(' {:<4d} {:<13s} {:<15d}'.format(l,key,
compare_one_value[m]))
            l+=1
        if l>word_input :
            break


def extraction_data(rotate_adress, book ,exit2) : # In there thanks
to requests and beautifulsoap code gets the book and write in .txt
file

        req = requests.get(rotate_adress + '/Print_version'  )
        if req.status_code == 200:
            soup = bs(req.content, 'lxml')
            file = open(book + ".txt", "w")
            for links in soup.find_all(class_="mw-parser-output"): #
retrieves all data in the given class
                file.write(links.text.encode('utf8').decode('ascii',
'ignore')) # Asci converts to encode because it can be a problem
            file.close()
        else : # if link has lower p code controls that and
continues
            req = requests.get(rotate_adress + '/print_version')
            if req.status_code == 200:
                soup = bs(req.content, 'lxml')
                file = open(book + ".txt", "w")
                for links in soup.find_all(class_="mw-parser-
output"):

file.write(links.text.encode('utf8').decode('ascii', 'ignore'))
                file.close()


while((manybooks !=1 or manybooks !=2) and exit == True) : #
continues until the user enters the correct number

    try : # If the user enters a wrong input, the code is not broken
and gives a warning message
        manybooks = int(input("\tPress 1 to learn about a book,
press 2 to compare between two books\n\t\tPress :  "))
        if(manybooks == 1) :

            first_book = input("\n\tThe title of the book you want
to learn\n\tName :  ").replace(" ", "_")
            print("\n\tThe number of words you want to find (Default
is 20)\n\tNumber :  ", end =" ")
            try : # If the user does not enter a value, it switches
to default setting.

                word_input = int(input())
            except :
                word_input = 20
                break
            break
        elif(manybooks==2) :
            first_book = input("\tThe name of the first book\n\tName
:  ").replace(" ", "_")
```

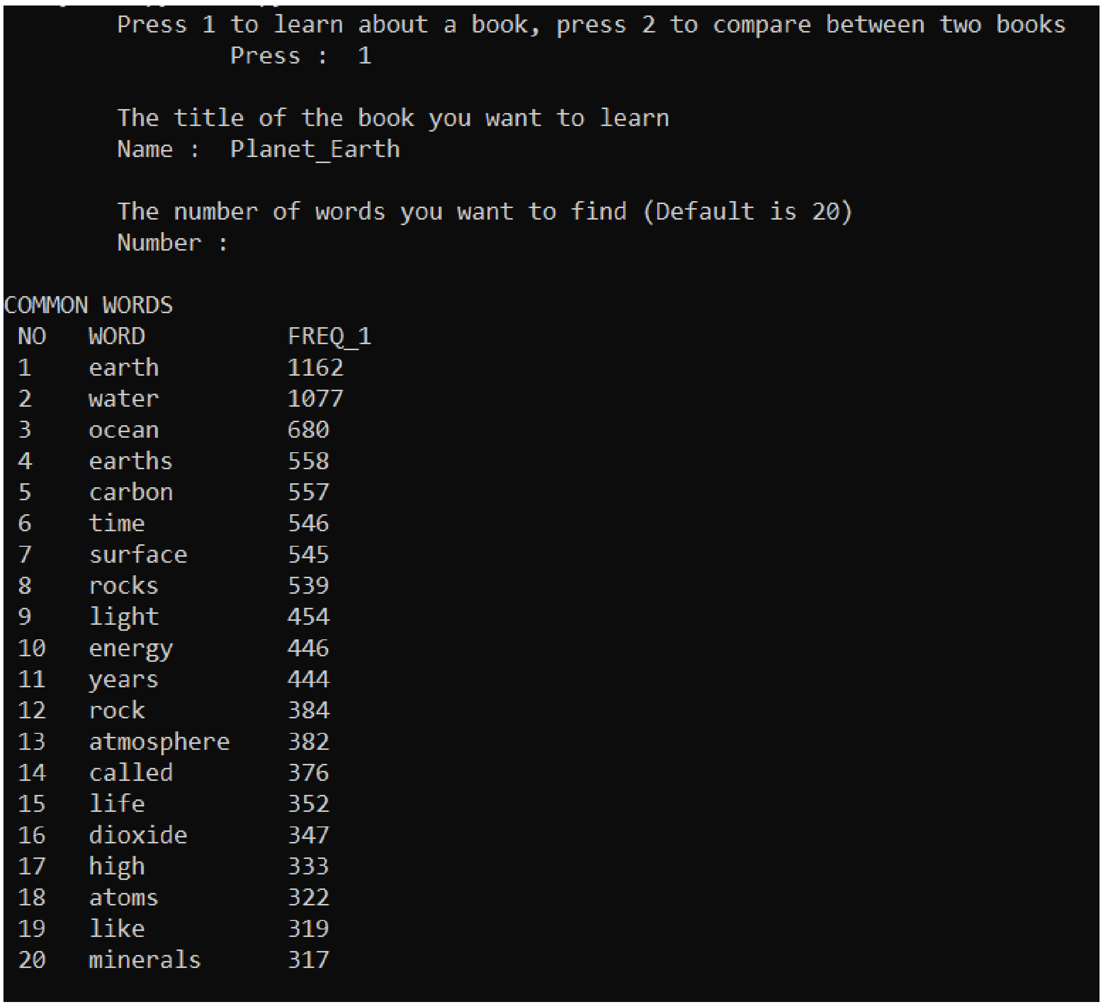
```
            second_book = input("\tThe name of the second
book\n\tName :  ").replace(" ", "_")
            print("\n\tThe number of words you want to find (Default
is 20)\n\tNumber :  ", end =" ")
            try:# If the user does not enter a value, it switches to
default setting.
                word_input = int(input())
            except:
                word_input = 20
                break
            break
        else : # If the user enters a wrong value, the code is not
broken and gives a warning message
            guide = ''
            while guide != 'a' or guide != 'A' or guide != 'E' or
guide != 'e' : # In there if user want to exit he/she can
                guide =  (input("\tYou cannot select more than 2 or
less than 1 book. Press 'E' to exit to 'A' to try again.\n\t\tPress
: "))
                if guide =='A' or guide == 'a':
                    break
                elif guide == 'E' or guide == 'e':
                    exit = False
                    break
                else :
                    print('\t\tYou pressed wrong key please
wait...')

    except (ValueError, ) :
        print('\n\t\tValue Error')
        print('\t\tWrong input please wait\n\n')
        print('-----------------------------------------------------
-------------------------')
        continue

if exit ==False :
    print('\t\tSee You ')
file = open(first_book + ".txt", "w").close()
rotate_adress = basic_adress + first_book
if(manybooks==1) : #if user entered 1
    s = 0
    try :
        extraction_data(rotate_adress, 'first_book', exit)# In there
thanks to requests and beautifulsoap code gets the book and write in
.txt file
        max_word_finder_for_dic('first_book' + '.txt',
firstbook_list_word_for_repetition) # find out how many times words
occur
        print('\nCOMMON WORDS\n', 'NO  ', "WORD\t", '   FREQ_1\t')
        just_first_book_print(keyfor1_stbook, valuesfor1_stbook) #
print
    except :
        print('\t\tThe book you are looking for is not available on
Wikisource\n\t\t See you.')
elif(manybooks ==2) : # if user entered 2
```

```
    rotate_adress2 = basic_adress + second_book
    s = 0
    try :
        extraction_data(rotate_adress, 'first_book', exit)
        max_word_finder_for_dic('first_book' + '.txt',
firstbook_list_word_for_repetition)
        s = 1
        extraction_data(rotate_adress2, 'second_book', exit)
        print('\nCOMMON WORDS\n', 'NO  ', "WORD\t", '   FREQ_1\t', "
FREQ_2\t", "   FREQ_SUM")
        max_word_finder_for_dic('second_book' + '.txt',
secondbook_list_word_for_repetition)
        print_for_double(keyfor2_ndbook, valuesfor2_ndbook)
        print('BOOK 1 : ',first_book,'\nDISTINCT WORDS\n', 'NO  ',
"WORD\t", '   FREQ_1\t')
        distinct(keyfor1_stbook, keyfor2_ndbook, valuesfor1_stbook)
        print('BOOK 2 : ',second_book,'\nDISTINCT WORDS\n', 'NO  ',
"WORD\t", '   FREQ_1\t')
        distinct(keyfor2_ndbook, keyfor1_stbook, valuesfor2_ndbook)
    except :
        print('\t\tThe book you are looking for is not available on
Wikisource\n\t\t See you.')
```

APPENDIX B: SCREENSHOTS OF YOUR USE CASES

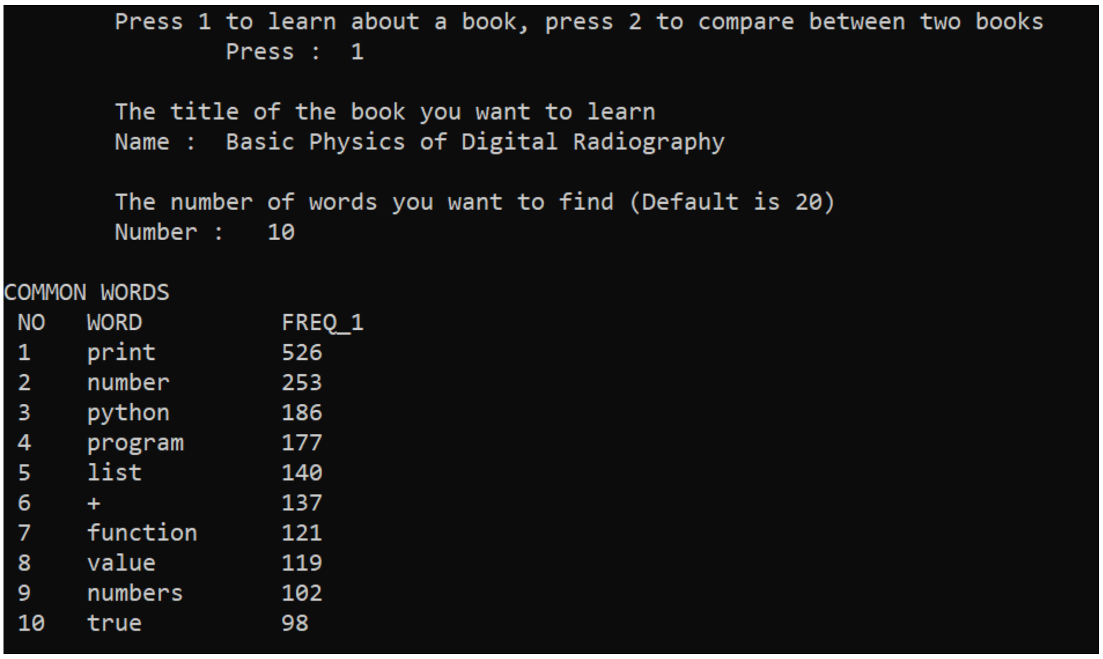


```
        Press 1 to learn about a book, press 2 to compare between two books
                Press :  1

        The title of the book you want to learn
        Name :  Planet_Earth

        The number of words you want to find (Default is 20)
        Number :

COMMON WORDS
 NO   WORD          FREQ_1
 1    earth         1162
 2    water         1077
 3    ocean         680
 4    earths        558
 5    carbon        557
 6    time          546
 7    surface       545
 8    rocks         539
 9    light         454
 10   energy        446
 11   years         444
 12   rock          384
 13   atmosphere    382
 14   called        376
 15   life          352
 16   dioxide       347
 17   high          333
 18   atoms         322
 19   like          319
 20   minerals      317
```

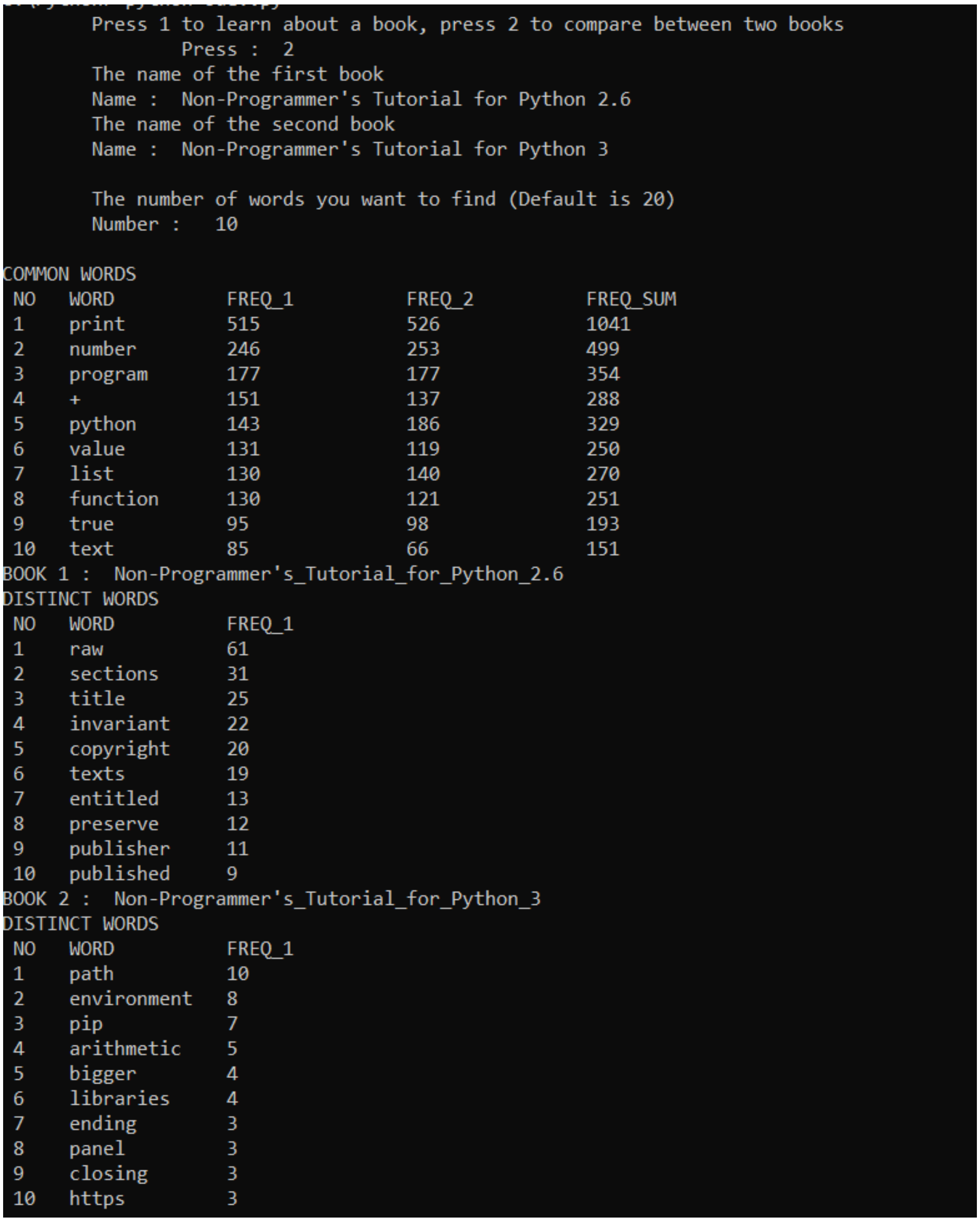


```
        Press 1 to learn about a book, press 2 to compare between two books
                Press :  1

        The title of the book you want to learn
        Name :  Basic Physics of Digital Radiography

        The number of words you want to find (Default is 20)
        Number :   10

COMMON WORDS
 NO    WORD          FREQ_1
 1     print         526
 2     number        253
 3     python        186
 4     program       177
 5     list          140
 6     +             137
 7     function      121
 8     value         119
 9     numbers       102
 10    true          98
```



```
        Press 1 to learn about a book, press 2 to compare between two books
                Press :  2
        The name of the first book
        Name :  Non-Programmer's Tutorial for Python 2.6
        The name of the second book
        Name :  Non-Programmer's Tutorial for Python 3

        The number of words you want to find (Default is 20)
        Number :   10

COMMON WORDS
 NO    WORD          FREQ_1          FREQ_2          FREQ_SUM
 1     print         515             526             1041
 2     number        246             253             499
 3     program       177             177             354
 4     +             151             137             288
 5     python        143             186             329
 6     value         131             119             250
 7     list          130             140             270
 8     function      130             121             251
 9     true          95              98              193
 10    text          85              66              151
BOOK 1 :  Non-Programmer's_Tutorial_for_Python_2.6
DISTINCT WORDS
 NO    WORD          FREQ_1
 1     raw           61
 2     sections      31
 3     title         25
 4     invariant     22
 5     copyright     20
 6     texts         19
 7     entitled      13
 8     preserve      12
 9     publisher     11
 10    published     9
BOOK 2 :  Non-Programmer's_Tutorial_for_Python_3
DISTINCT WORDS
 NO    WORD          FREQ_1
 1     path          10
 2     environment   8
 3     pip           7
 4     arithmetic    5
 5     bigger        4
 6     libraries     4
 7     ending        3
 8     panel         3
 9     closing       3
 10    https         3
```

REFERENCES

For stop words

1 – Wikipedia

https://en.wikipedia.org/wiki/Stop_word Rajaraman, A.; Ullman, J. D. (2011).

2- xpo6

http://xpo6.com/list-of-english-stop-words/ January 25, 2015

For Requests module

1- https://requests.readthedocs.io/en/master/ Kenneth Reitz
2- https://pypi.org/project/requests/ Dec 16, 2020 Kenneth Reitz
3- https://www.w3schools.com/python/module_requests.asp

For Beutifulsoup module

1- https://pypi.org/project/beautifulsoup4/ Oct 3, 2020 Leonard Richardson
2- https://medium.com/@nuriyavuz2.71/python-beautifulsoup-modülü-689ef499ee16 Jan 18 2020 Nuri Yavuz
3- https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser)
   Leonard Richardson 2004

For using python elements (Dictionary , list)

1- https://www.w3schools.com/python/python_dictionaries.asp
2- https://realpython.com/python-dicts/ John Sturtz
3- https://docs.python.org/3/tutorial/datastructures.html
4- https://www.w3schools.com/python/python_lists.asp

Books

Starting out with Python (Global Edition) Tony Haddis 2008