**NAME: VANSHAJ SHARMA**

**ROLLNO: MT23103**

**COURSE: M.TECH CSE**

**YEAR: FIRST YEAR**

**Q1**

**Approach :**

In this Question, I used libraries such as nltk, pathlib, bs4 and os. Nltk is used to provide functions for natural language processing like word tokenizer, stemming and lemmatization etc. For this question I performed following steps in the following order.

a. Lowercase the text

b. Perform tokenization

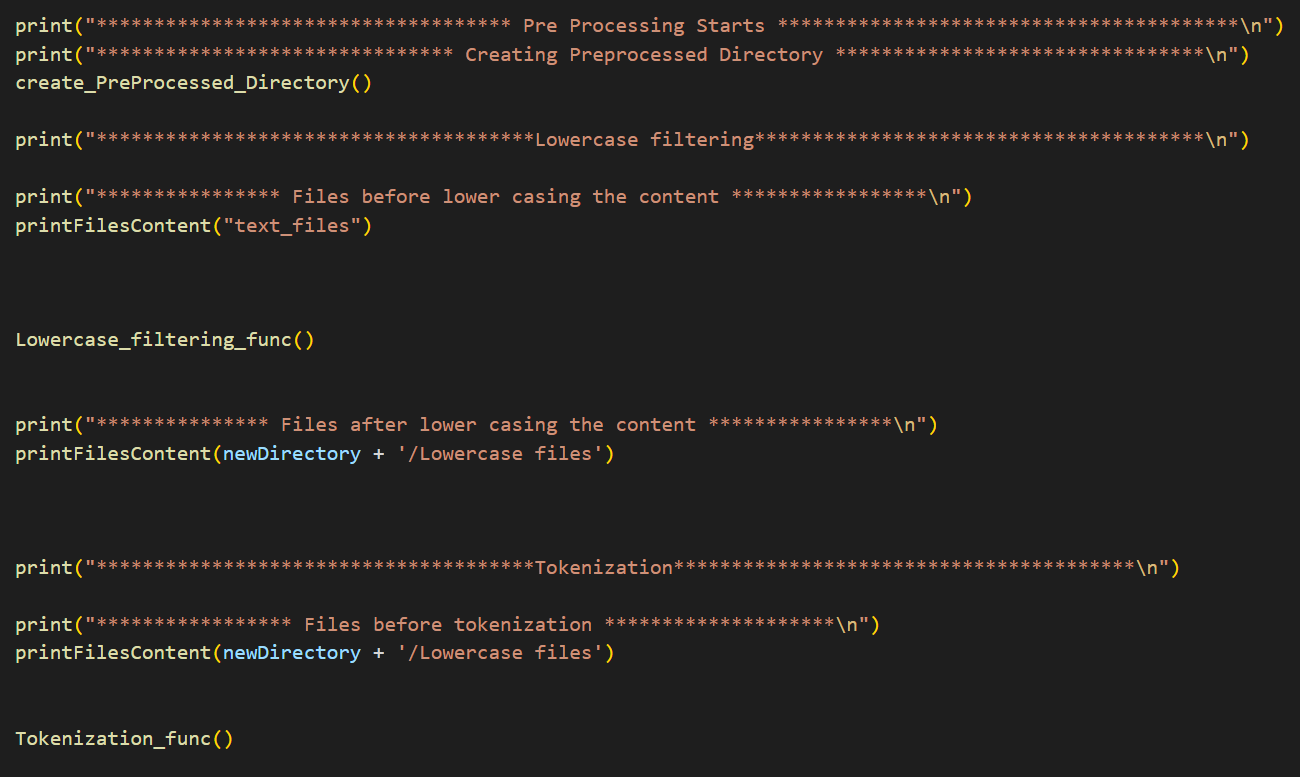
c. Remove stopwords

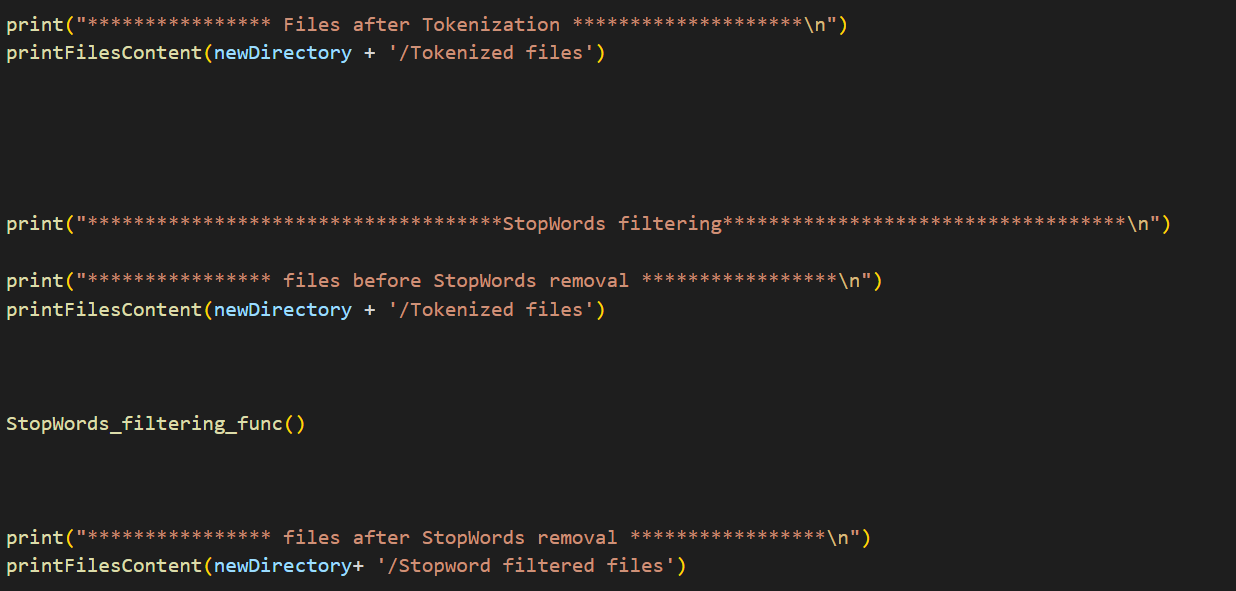
d. Remove punctuations

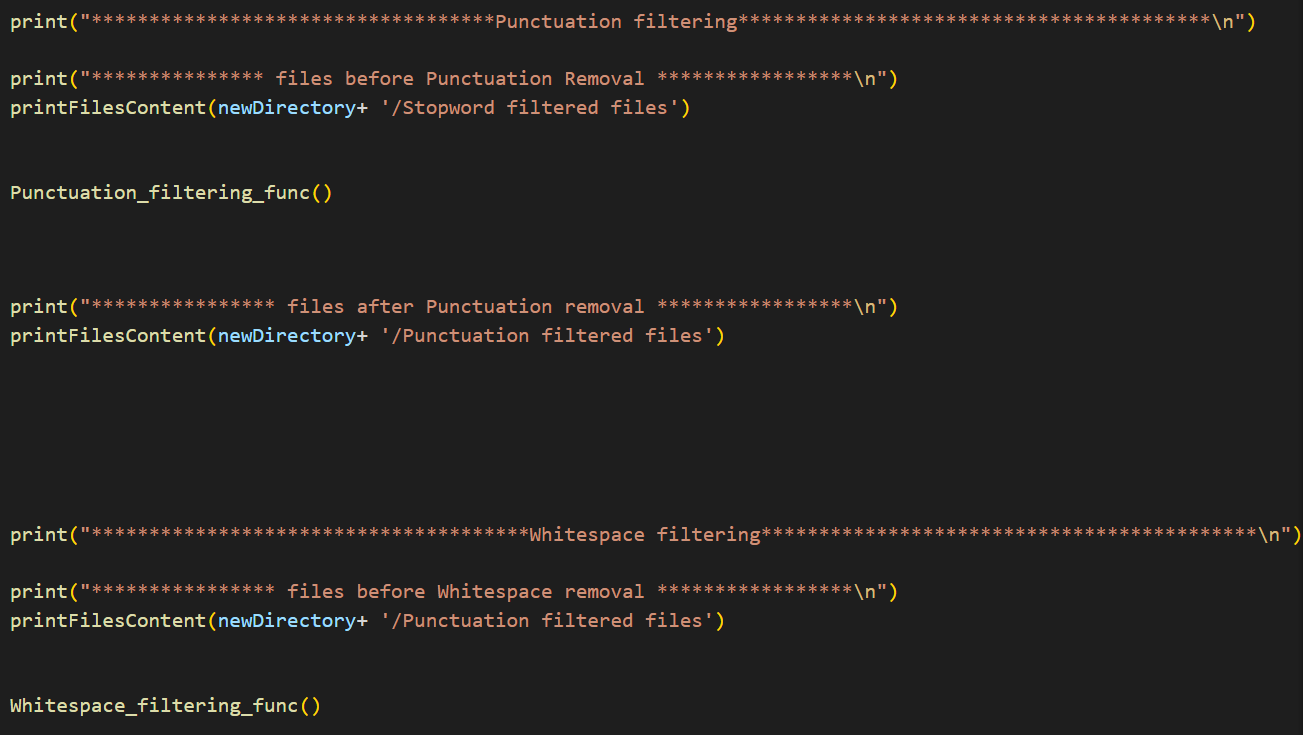
e. Remove blank space tokens

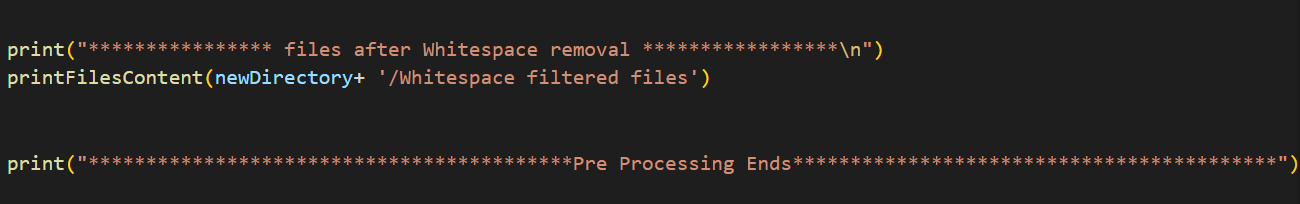
After each step, I am saving resultant files in the corresponding folder to these functions. Here each step is taking input the output of previous step and then processing giving the results.

Before and after each step, we are printing all the files so that we could be able to see the changes have been done by each function. Order of processing is :-

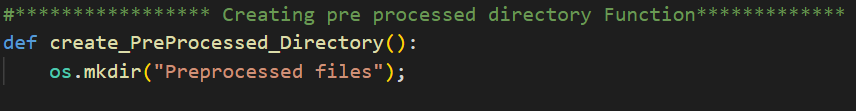


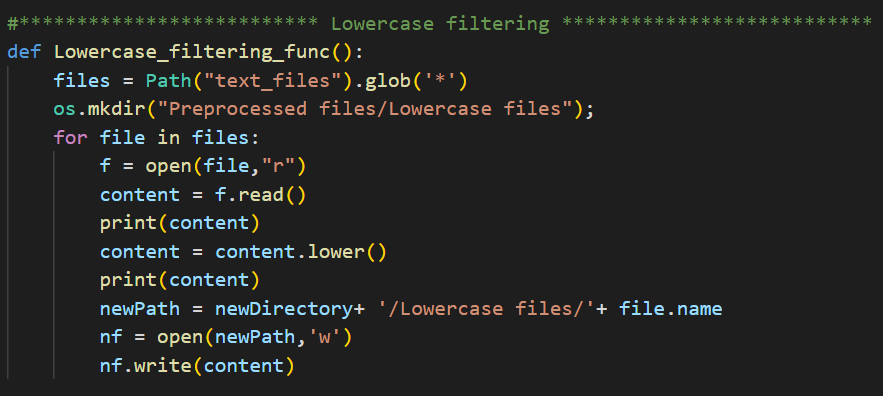


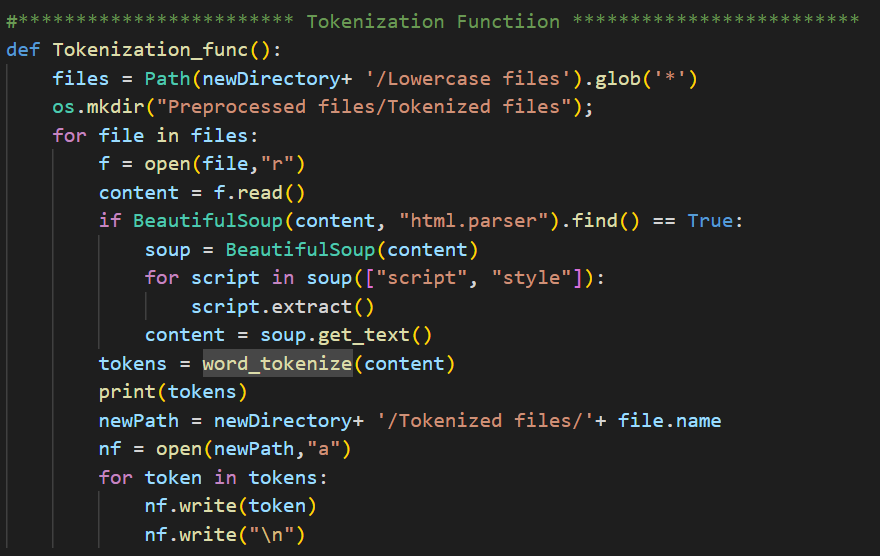




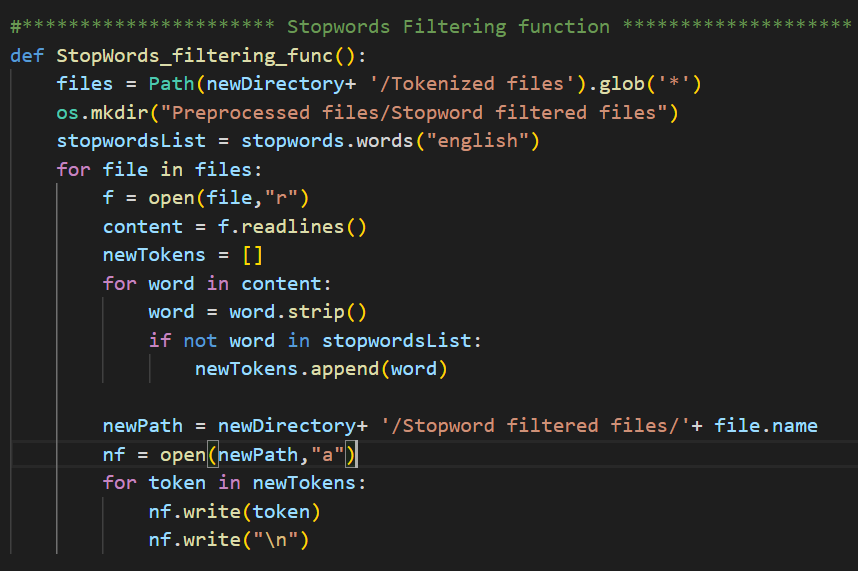
**Methodologies :**

**First , I am creating a directory called pre processed files. In which the resultant folder for each pre processing step will be there.**

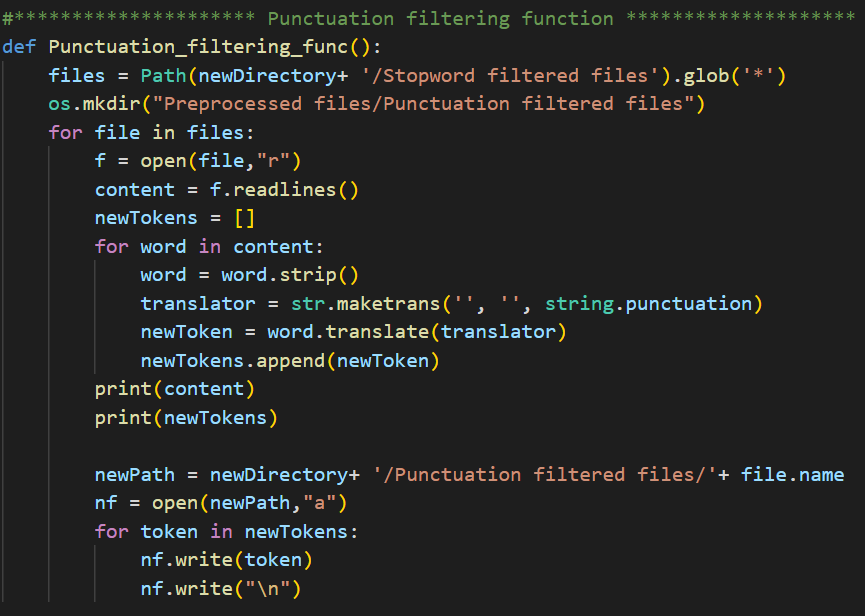
**a. Lowercase the text:** I created a function of this step, in which I first created the directory called lowercase files inside the pre-processed files directory then I am reading all the 999 files given in the assignment and converting their content in lowercase and saving the content in the new 999 files inside the lowercase directory.

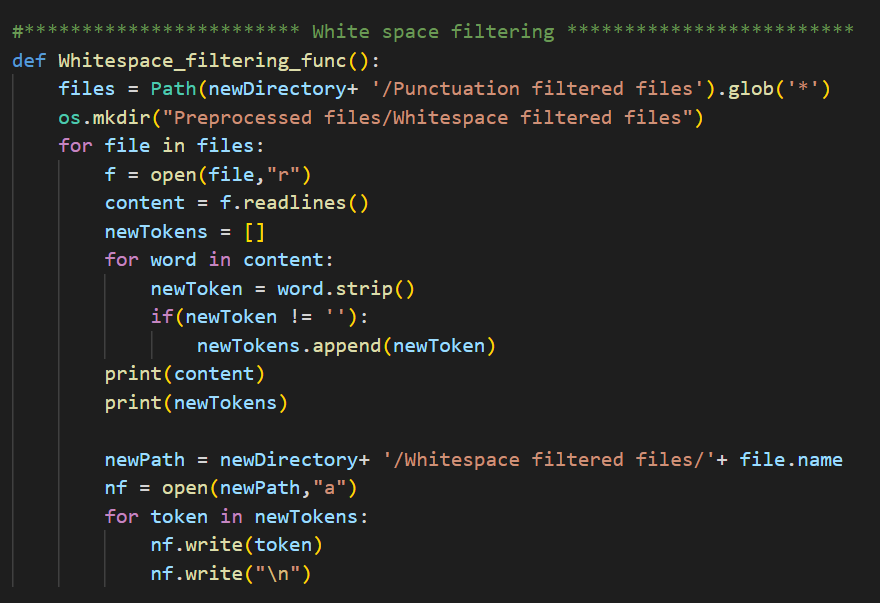
**b. Perform tokenization:** In this function, I created a directory called Tokenized files. In which I am storing the 999 files which are having tokens of the text in 999 files that we produced for previous step i.e lowercase filtering. I used function word\_tokenize from nltk library for tokenization.

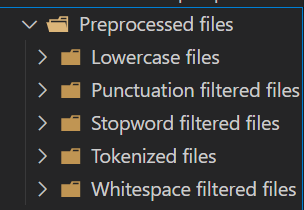
**c. Remove stopwords:** In the tokens that we generated for each file of 999 files. There are some tokens that are irrelevant that are not providing much meaning to the text. I created a directory for this step called stop word filtering files . So, using stop words list provided by the nltk library, I will parse each file among 999 files from tokenization directory and create corresponding new file in the stop word filtered files directory by saving all the tokens that are not in stop word list.



**d. Remove punctuations:**  For this step, I created a directory called punctuation filtered files. In which I stored the new files with the content of files in stop word filtering files after removing punctuation from these files. For removing punctuation, I used String class’s translator and maketrans function with parameter “”,”” in it.



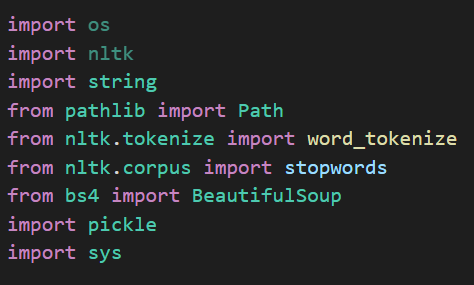
**e. Remove blank space tokens:** For performing this step, I created a new folder for this step called Whitespace filtered files. I used resultant files from previous step and removed white space from the tokens list in these files and created new files from the tokens as content without white space in white space directory. For this I just checked whether the this token is whitespace or not. 

**Results:**

**CLI output is very large so can’t mention it here, please run the Q1 file and see the output.**

**Q2**

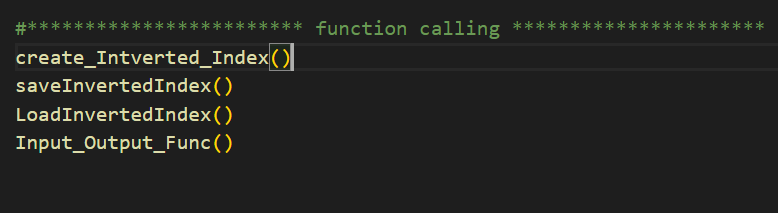
**Approach:**

**Libraries used**

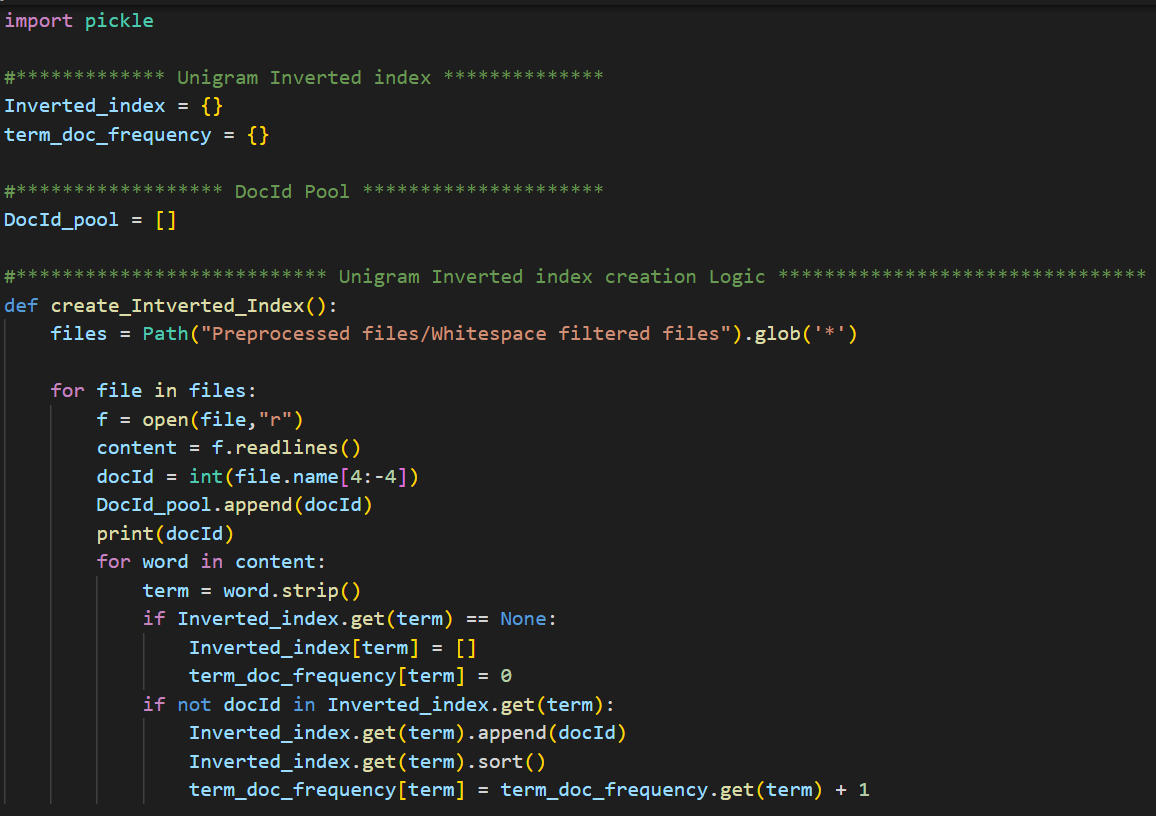
This question is about to create inverted index , So I created inverted index from the tokens that are generated for each file in question 1. I considered the token in files as terms and the file no is document id. Here I am making dictionary of key and value pair . In which key is the term and value is list of document id in which key appears. for performing operations on Query like AND, OR, AND NOT and OR NOT. I created functions that receive lists in the argument and then this function performs intersection or function related thing to find answer and return the list of documents id for the query. Q2 program take input from user the Query and the operations in needed to be performed on the query then it will preprocess the Query with the same method as we did in Question 1. We will process the query in the left to right in terms of operators. Used pickle library to save the inverted index into byte format and load the byte format inverted index into dictionary format inverted index.

**Methodologies:**

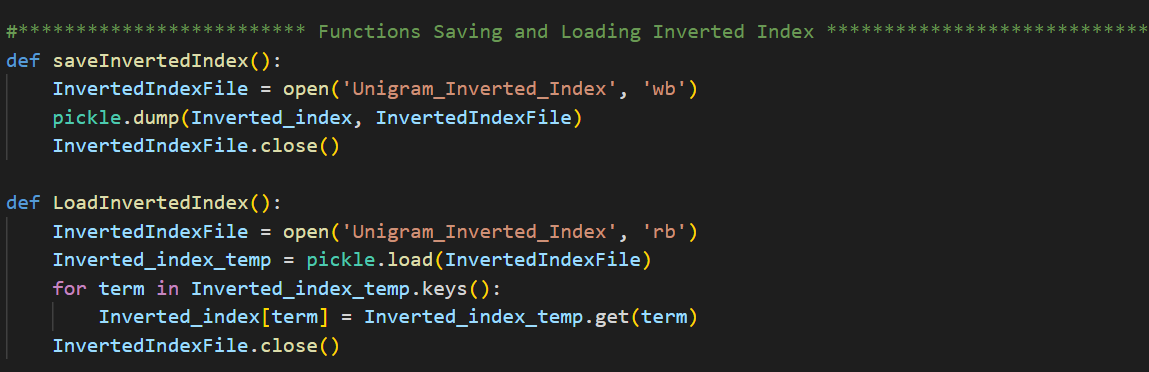
**Order of execution of the function**

****

**This function create\_Inverted\_Index will create inverted index using dictionary. In this function we are traversing documents one by one only creating new term as key in dictionary if it is not already present as key and if we are creating key for the first time then we initialize the list and add document id in it else if the term already there in the dictionary as key then we will add document id in the list of document ids in the value portion of that term as key. Whenever I talked about dictionary means that inverted index**.

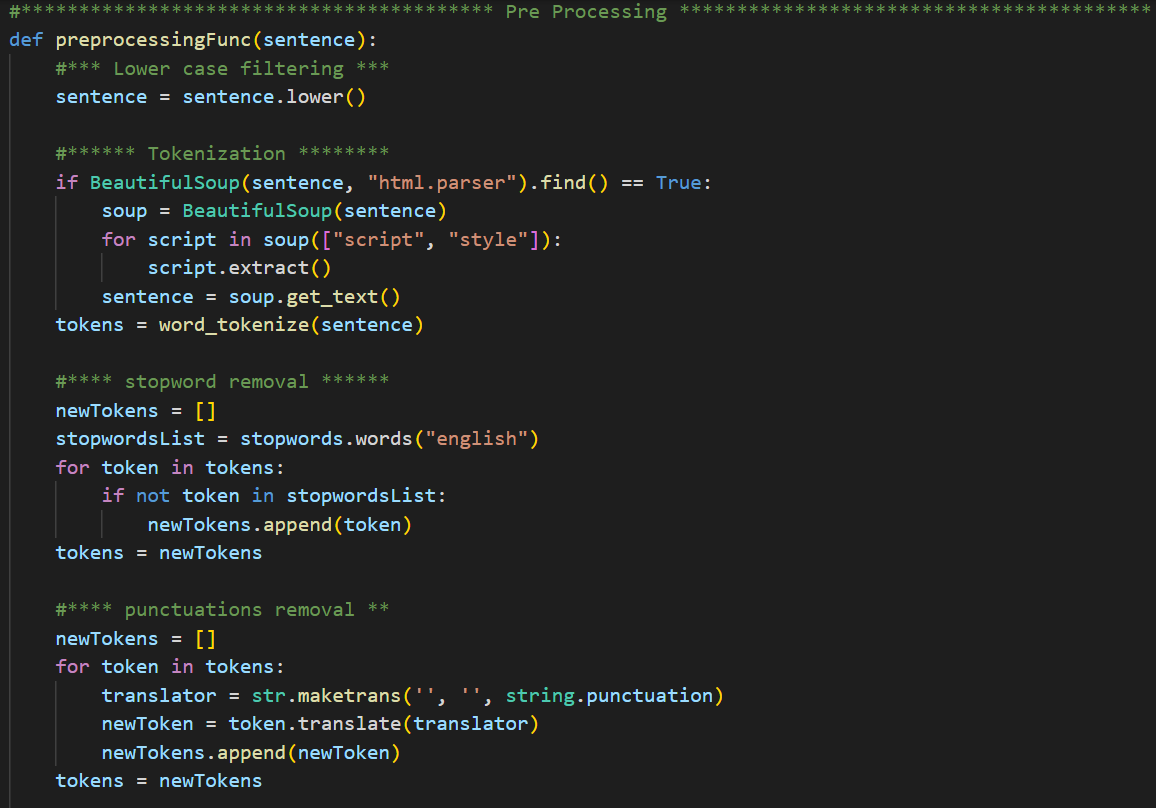
****

**Save inverted index Function for storing inverted index into byte format and Load inverted index function for loading stored byte format inverted index into python dictionary.**

****

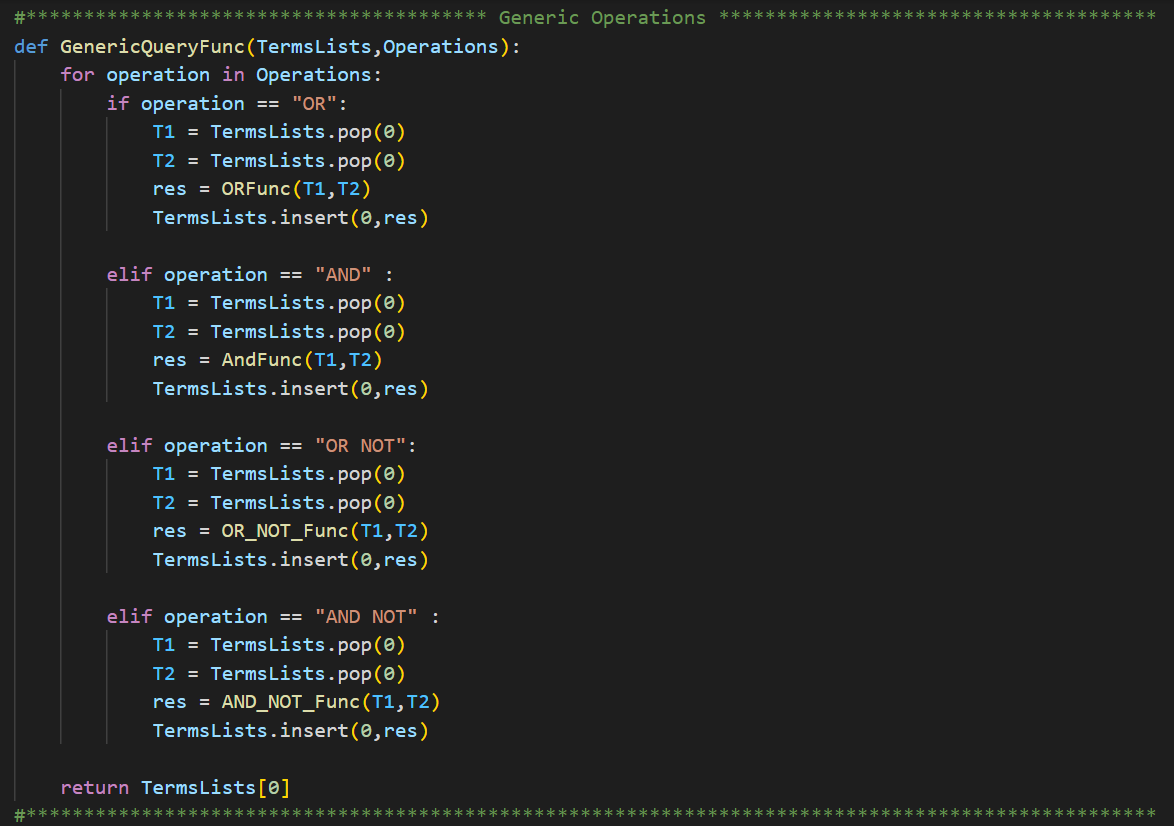
****

**In the below function we are doing preprocessing for the Query got through the input. We folled same steps for preprocessing as we performed in Question 1**

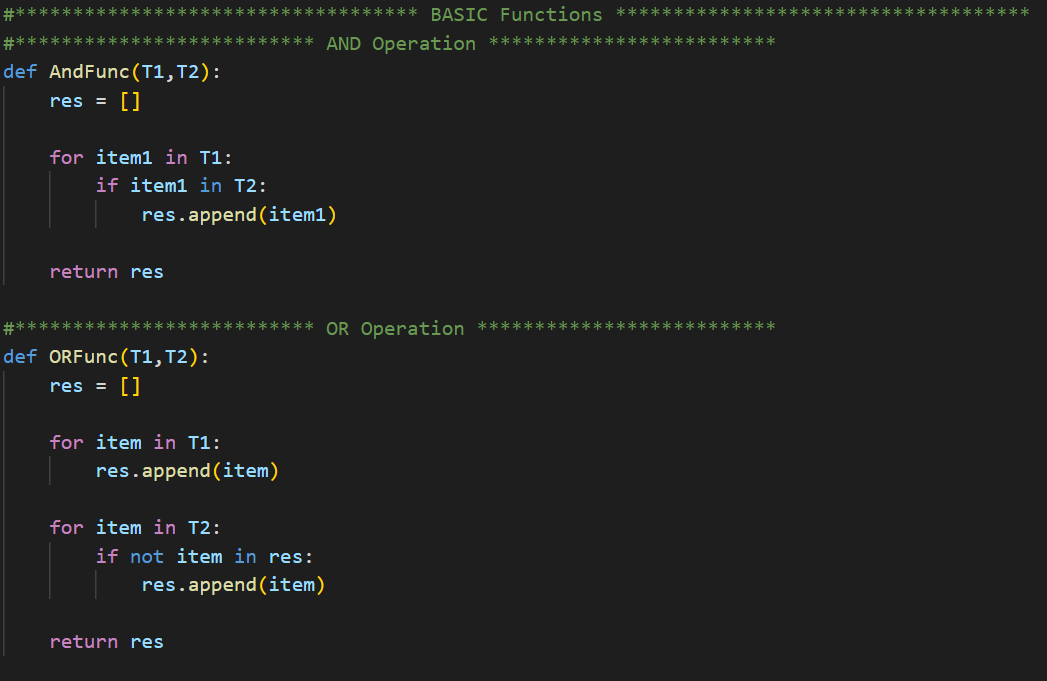
****

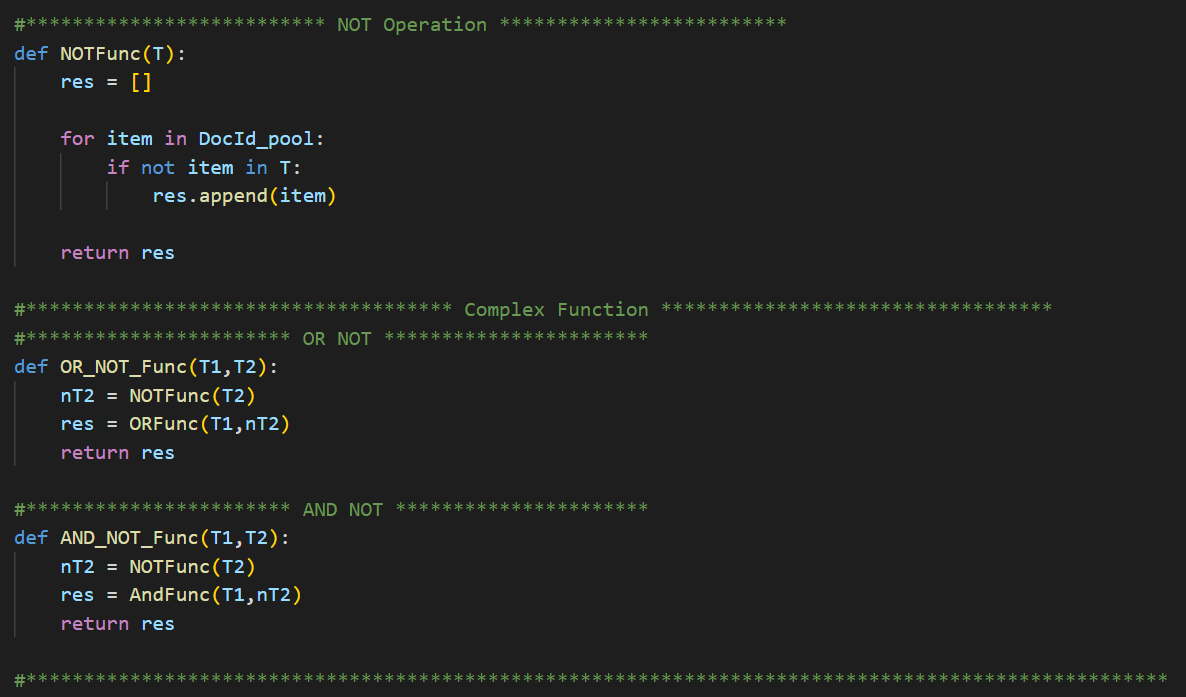
****

**This function GebericQueryFunc() is for processing Query’s terms with operations given. In this we are processing terms in the Query in left to right fashion with respect to operations received as input. In this function we will get two lists one is list of document id’s lists associated to each term mentioned in the Query in the same order of the terms in Query and second one is list of operations which is formed after splitting input operations by delimeter ‘,’. Here we have handled cases for four types of operations AND , OR, AND NOT, OR NOT. In our function loop is looping for no of operations times for each operation we are performing operations on first two list of documents id after removing them list of lists of document id and after performing operations we are adding its result bask to first position of the list of lists of document id. At last returning left over list of doc id in list of lists of Document id**

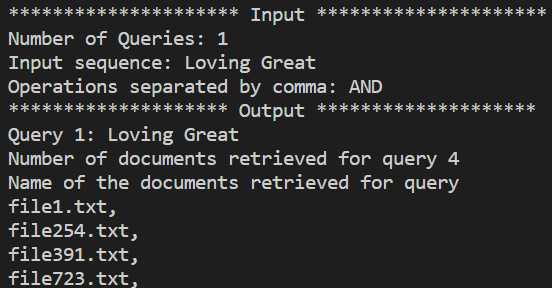
****

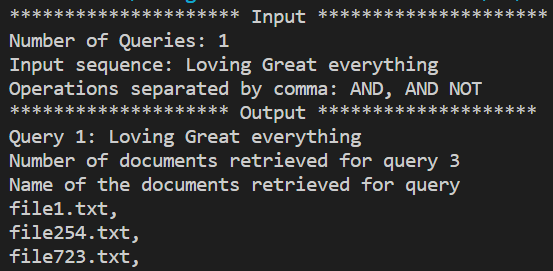
**These below are the basic functions that are binary operations like AND , OR , AND NOT , OR NOT. These functions take input two list of document Ids and perform corresponding operations.**

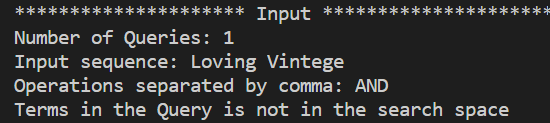
****

****

**Results:**

****

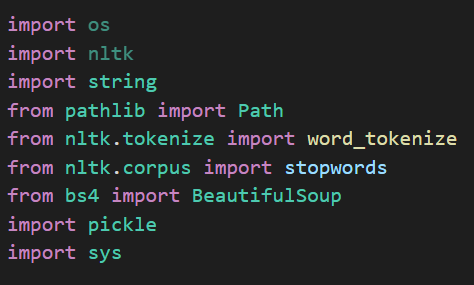
****

****

**Q3**

**Approach:**

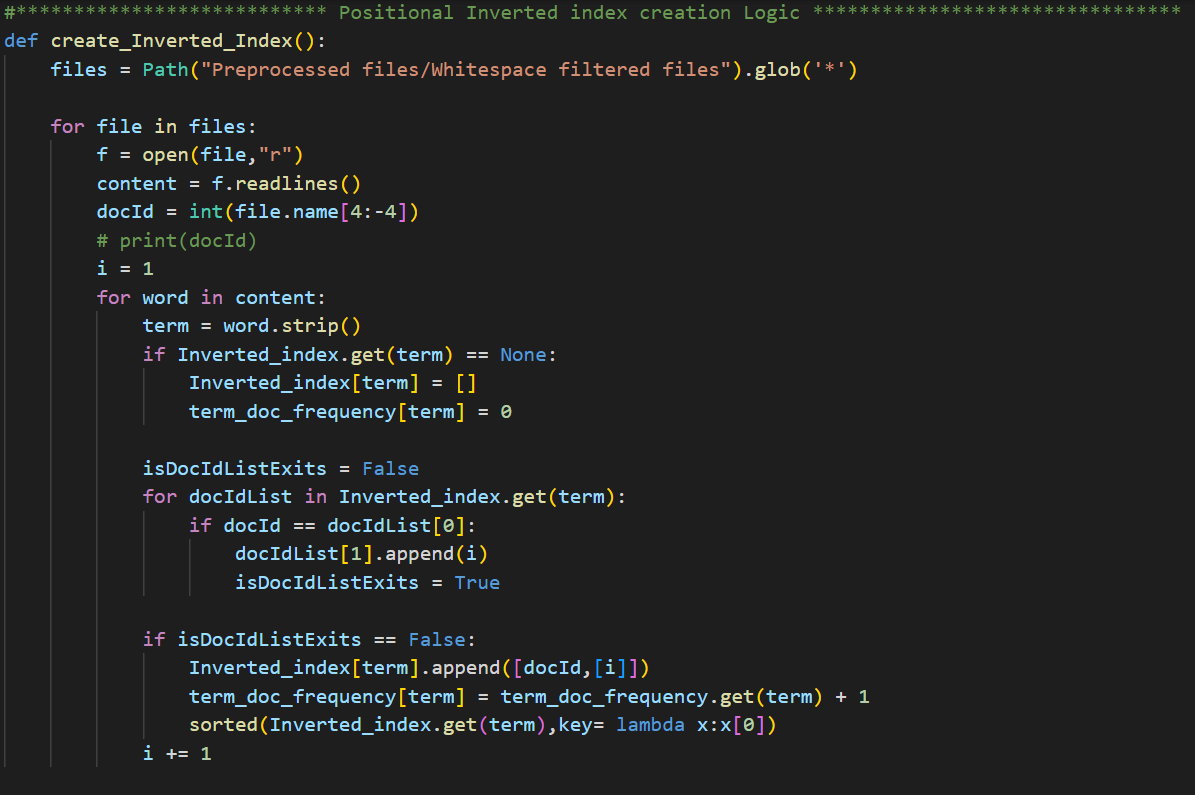
**Libraries used:**

****

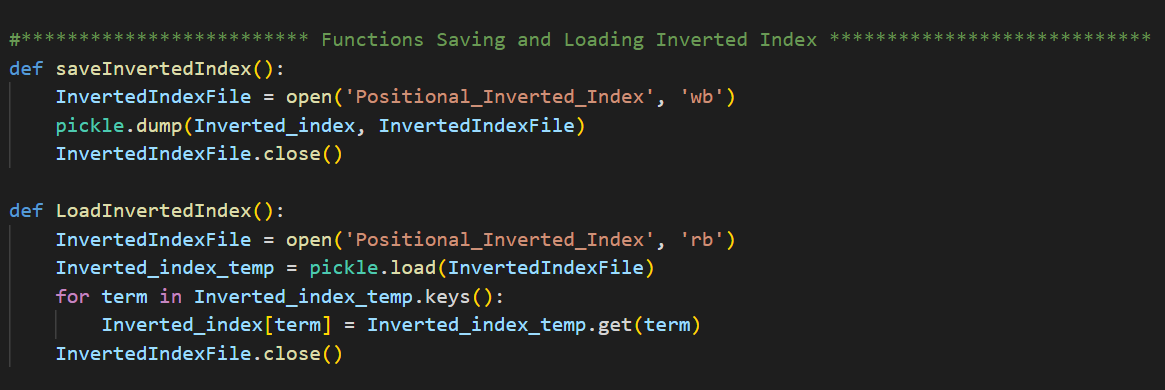
In this we created inverted index same as Question 2 but in this we also maintained the list of positions of each term inside the each document. In this we used find contained documents to find which document are having the terms in query in the same sequence as of query. Our approach for finding the documents that are having same sequence of query and its is recursive . In this each take first term of the query and check in its corresponding documents whether the another terms are present in the same document or not in same sequence position if not then this document will not be in answer else this document will be in answer.

**Methodologies:**

**This function below create the positional inverted index after reading all the files.**

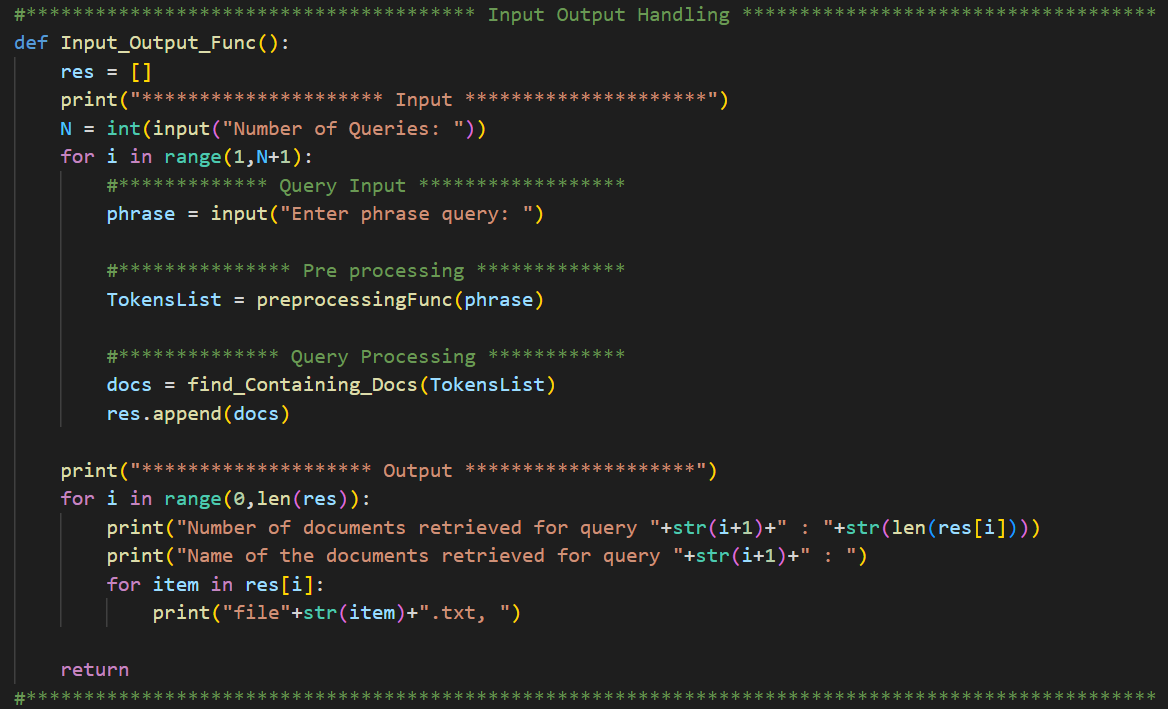
****

**The function saveinvertedindex() is used to save positional inverted index into byte format and loadInvertedIndex() is used to load the byte format positional inverted index to python dictionary format positional inverted index.**

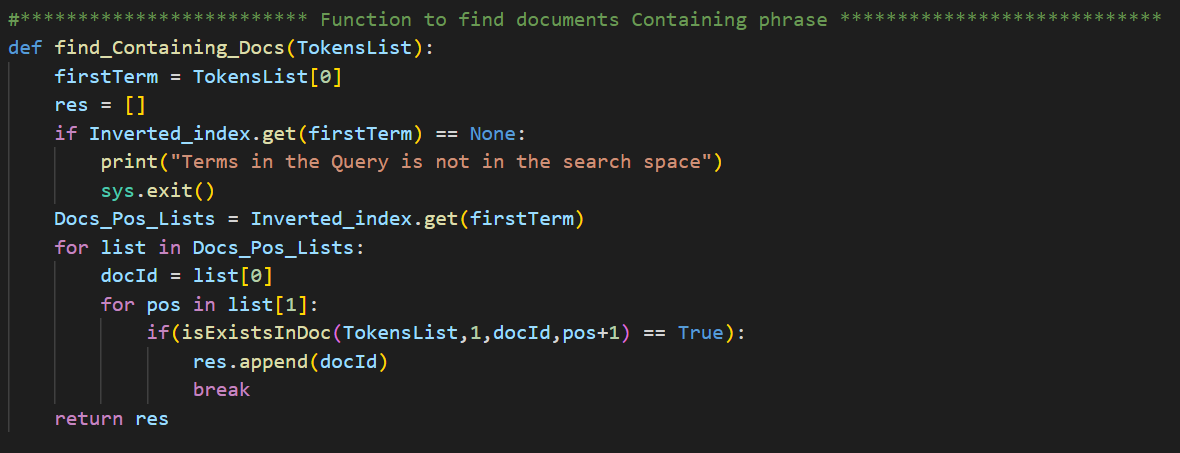
****

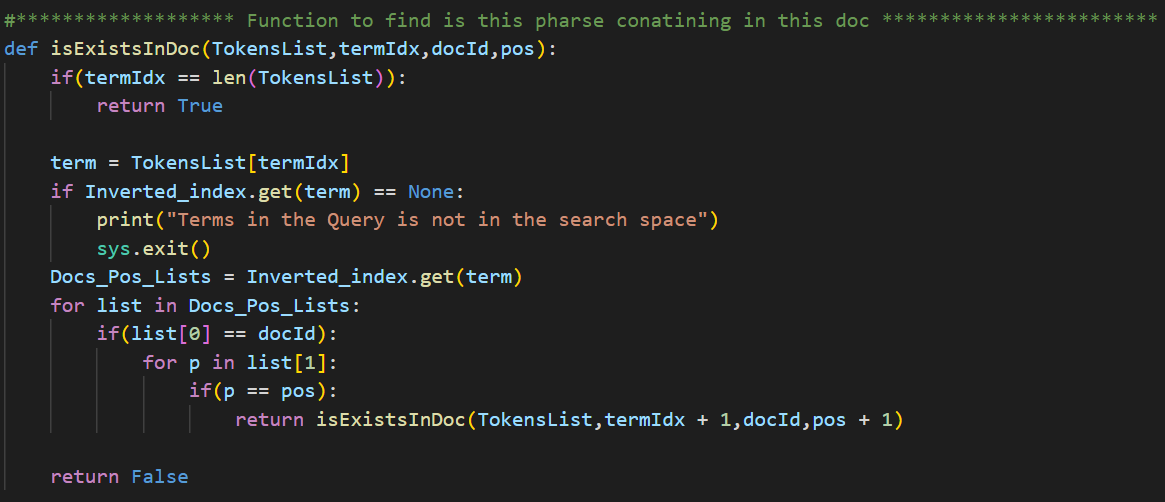
****

**This Input\_output\_Func() takes input all the queries and process and print all the results of the queries.**

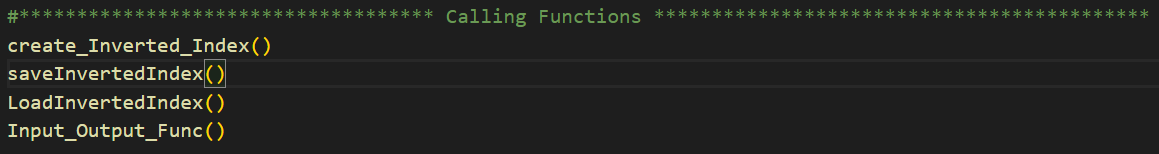
****

**These below two helper functions are used to find the documents that are having query terms inputted by user in the same sequence as mentioned in query.**

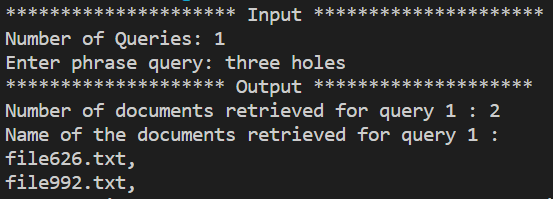
****

****

**Order of function execution**

****

**Results:**

****