```
import numpy as np
import pandas as pd
def check_parenthesis(s):
    l=[]
    for i in s:
        if i=='(':
            l.append(i)
        elif i==')' and len(l)>0:
            l.pop()
        else:
            continue
    if len(l)==0:
        return True
    else:
        return False

def Precedence(a):
    if a.lower()=='and': return 1
    elif a.lower()=='or': return 2
    elif a.lower()=='not': return 3
    else:return 0

def Operator(a):
    if a.lower() in ['(',')','and','or','not']: return True
    else: return False

def infix_to_postfix(s):
    operators=[]
    operands=[]
    for i in s:
        if i=='(': operators.append(i)
        elif i==')':
            while(len(operators)!=0 and operators[-1]!='('):
                operands.append(operators.pop())
            operators.pop()
        elif not Operator(i):
            operands.append(i)
        else:
            while(len(operators)!=0 and
Precedence(i)<=Precedence(operators[-1])):
                operands.append(operators.pop())
            operators.append(i)
    while(len(operators)!=0):
           operands.append(operators.pop())
    return operands

def queryPrint(r):
    if len(r)!=0:
        print('The Documents related to the given query are:')
        for i in r:
```

```
            print(i)
    else:
        print('No relavent documents found')

def queryEval(s,d,dn):
    if(check_parenthesis(s)):
        s=infix_to_postfix(s)
        operands=[]
        for i in s:
            if not Operator(i):
                operands.append(i)
            elif i.lower()=='and':
                b=operands.pop()
                res=AND_OF(operands.pop(),b,d)
                operands.append(res)
            elif i.lower()=='or':
                b=operands.pop()
                res=OR_OF(operands.pop(),b,d)
                operands.append(res)
            else:
                res=NOT_OF(operands.pop(),d,dn)
                operands.append(res)
        res=operands.pop()
        print(res)
        queryPrint(res)
    else:
        print('Invalid no of Parenthesis in the given query.')
        return

def AND_OF(a,b,d):
    if type(a) is str:
        l1=d[a]
    else:
        l1=a
    if type(b) is str:
        l2=d[b]
    else:
        l2=b
    r=[]
    for i in l1:
        if i in l2:
            r.append(i)
    return r

def OR_OF(a,b,d):
    if type(a) is str:
        l1=d[a]
    else:
        l1=a
    if type(b) is str:
```

```
        l2=d[b]
    else:
        l2=b
    for i in l1:
        if i not in l2:
            l2.append(i)
    return l2

def NOT_OF(a,d,dn):
    if type(a) is str:
        l=d[a]
    else:
        l=a
    r=[]
    for i in dn:
        if i not in l:
            r.append(i)
    return r

#docNames=['Doc1','Doc2','Doc3','DOC4']
docNames=['DOC1','DOC2','DOC3','DOC4']
docs=[]
for i in docNames:
    try:
        with open(f'{i}.txt') as file:
            data=file.readlines()
            docs.extend(data)
    except:
        print(f'There is a Problem opening the file {i}.txt')

for i in range(len(docNames)):
    print(f'{docNames[i]}:\n{docs[i]}')

docTerms=[]
for i in docs:
    for j in i.lower().strip().split():
        if j not in docTerms:
            docTerms.append(j)
docTerms.sort()

dic={}
for i in range(len(docTerms)):
    l=[]
    for j in range(len(docs)):
        if docTerms[i].lower() in docs[j].lower():
            l.append(docNames[j])
    dic[docTerms[i]]=l

#print('\nTerms\t    Docs')
#print('----------------')
```

```
#for i in Inv_ind:
#    print(i,'-----',Inv_ind[i])

s=input('\nEnter your query:').split()
queryEval(s,dic,docNames)
#look and even and slightly
```

```
DOC1:
It is a long established fact that a reader will be distracted by the
readable content of a page have suffered when looking at its layout
DOC2:
Many desktop publishing packages and web page editors have suffered
alteration now use Lorem Ipsum as their default model text, and a
search for lorem ipsum will uncover many web sites still in their
infancy
DOC3:
There are many variations of passages of Lorem Ipsum available but the
majority have suffered alteration in some form by injected humour or
randomised words which donot look even slightly believable
DOC4:
But I must explain to you how all this mistaken idea of denouncing
pleasure and praising pain was born and I will give you a complete
account of the system and expound the actual teachings of the great
explorer of the truth the master builder of human happiness

Enter your query:systems and powered and ai

---------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
Cell In[48], line 36
     30 #print('\nTerms\t    Docs')
     31 #print('----------------')
     32 #for i in Inv_ind:
     33 #    print(i,'-----',Inv_ind[i])
     35 s=input('\nEnter your query:').split()
---> 36 queryEval(s,dic,docNames)
     37 #look and even and slightly

Cell In[43], line 10, in queryEval(s, d, dn)
      8 elif i.lower()=='and':
      9     b=operands.pop()
---> 10     res=AND_OF(operands.pop(),b,d)
     11     operands.append(res)
     12 elif i.lower()=='or':

Cell In[44], line 3, in AND_OF(a, b, d)
      1 def AND_OF(a,b,d):
      2     if type(a) is str:
```

```
----> 3         l1=d[a]
      4     else:
      5         l1=a

KeyError: 'systems'
```