positional indexes

```
doc4="'hello its information retrieval A paragraph is a series of sentences that are organized an
d coherent, and are all related to a single topic. Almost every piece of writing you do that is long
er than a few sentences should be organized into paragraphs.
       system contains boolean and phrasal queries'"
print(doc4)
doc5="'Information retrieval (IR) in computing and information science is the process of obtainin
g
      information system resources that are relevant to an information need from a collection of th
ose resources. Searches can be based on full-text or other content-
based indexing. Information retrieval is the science[1] of searching for information in a document
, searching for documents themselves, and also searching for the metadata that describes data,
 and for databases of texts, images or sounds'"
print(doc5)
doc6="'An information retrieval process begins when a user enters a query into the system. Que
ries are formal statements of information needs, for example search strings in web search engin
es.
 In information retrieval a query does not uniquely identify a single object in the collection.
 Instead, several objects may match the query, perhaps with different degrees of relevance.'"
print(doc6)
l2=[doc4.lower(),doc5.lower(),doc6.lower()]
k=[]
p=l2[0].split(' ')
#print(p)
for i in p:
  k.append(i.strip())
#print(k)
r=[]
# Remove multiple empty spaces from string List
# Using loop + strip()
for i in range(0,len(l2)):
  z=l2[i].split(" ")
  res = []
  for ele in z:
    if ele==":
      continue
    else:
       k=ele.strip()
       res.append(k.lower())
  r.extend(res)
print(r)
u=[]
for i in r:
  if  i.lower()  not in stopwords and i.upper() not in stopwords and i not in stopwords:
       u.append(i)
print(u)
u=sorted(u)
print(u)
l3=[doc4.lower(),doc5.lower(),doc6.lower()]
d1={}
```

```
for i in u:
  d1.update({i:{}})
  for j in range(0,len(l2)):
      phi=l2[j].split(" ")
      #print(phi)
      if(i in phi):
           d1[i].update({j:[]})
           c_ount=phi.count(i)
           print(c_ount)
           for k in range(0,c_ount):
               ind_ex=phi.index(i)
               d1[i][j].append(ind_ex)
               phi.remove(i)

print(d1)
def and_oper(pp1,pp2):
  if(len(pp1)==0):
     return pp2
  elif (len(pp2)==0):
    return pp1
  res=[]
  z1=set(pp1)
  z2=set(pp2)
  ut=z1.intersection(z2)
  return list(ut)

def getpositional(l):
  rz=[]
  if(len(l)==1):
    rz=d1[l[0]].keys()
    return rz
  rz=and_oper(list(d1[l[0]].keys()),list(d1[l[1]].keys()))
  for i in range(2,len(l)):
    rz=and_oper(rz,list(d1[l[i]].keys()))
  return rz


def checkcone(l):
  flag=1
  for i in range(0,len(l[0])):
       rt=l[0][i]
       for j in range(1,len(l)):
         for dr in range(0,len(l[j])):
              if(rt+1==l[j][dr]):
                flag+=1
       if(flag==len(l)):
          return flag
```

```
s3=input("enter the query to be processed")
print(s3)
s3=input("enter the query to be processed")
print(s3)
#read the postings list of phrasal queries
k=s3.split(" ")
f=getpositional(k)
print(f)
answer=[]
for i in f:
  pes=[]
  for j in k:
    pes.append(d1[j][i])
  #print(pes)
  resu=checkcone(pes)
  #print(resu)
  if(resu==len(k)):
    print("present in document",(i))
    answer.append(i)
  else:
    print("not present in document",(i))
print(answer)
[2]
present in document 2
[2]
```