THE OBJECTIVE IS TO CREATE CLUSTERS OF 'SIMILAR' ARTICLES WITHIN A CORPUS OF ARTICLES FROM 1 BLOG

1. CREATE A CORPUS OF BLOG POSTS BY DOWNLOADING ALL THE ARTICLES FROM A BLOG

THIS WILL INVOLVE PARSING THE HTML TO REMOVE ALL THE CRUD (DIVS/TAGS)

2. USE THE K-MEANS CLUSTERING ALGORITHM TO IDENTIFY 5 CLUSTERS OF ARTICLES

2. USE THE K-MEANS CLUSTERING ALGORITHM TO IDENTIFY 5 CLUSTERS OF ARTICLES

USE TF-IDF TO REPRESENT
EACH ARTICLE AS A VECTOR

INITIALIZE A SET OF MEANS (CENTROIDS
OF THE 5 CLUSTERS TO BE FOUND)

ASSIGN EACH DOCUMENT/ARTICLE TO
THE NEAREST MEAN

ONCE THIS IS DONE COMPUTE THE CENTROIDS
OF THE CLUSTERS THAT ARE FOUND AND MAKE
THEM THE NEW MEANS

K-MEANS WILL CONVERGE WHEN THE
ASSIGNMENTS NO LONGER CHANGE
(IN OUR CASE WE WILL STOP AFTER A
CERTAIN NUMBER OF ITERATIONS)

WE WILL USE

SCIKIT-LEARN

AN OPEN SOURCE LIBRARY
FOR MACHINE LEARNING IN PYTHON

CREATE A CORPUS OF BLOG POSTS

```
def getAllDoxyDonkeyPosts(url,links):
    request = urllib2.Request(url)
    response = urllib2.urlopen(request)
    soup = BeautifulSoup(response)
    for a in soup.findAll('a'):
        try:
            url = a['href']
            title = a['title']
            if title == "Older Posts":
                print title, url
                links.append(url)
                getAllDoxyDonkeyPosts(url,links)
        except:
            title = ""
    return

blogUrl = "http://doxydonkey.blogspot.in"
links = []
getAllDoxyDonkeyPosts(blogUrl,links)
doxyDonkeyPosts = {}
for link in links:
    doxyDonkeyPosts[link] = getDoxyDonkeyText(link,'post-body')

documentCorpus = []
for onePost in doxyDonkeyPosts.values():
    documentCorpus.append(onePost[0])
```

GET ALL LINKS FROM THE DOXYDONKEY BLOG (USE BEAUTIFUL SOUP TO FIND THE URLS WITHIN THE BLOG'S WEBPAGE)

FOR EACH LINK, DOWNLOAD AND PARSE THE WEBPAGE TO GET THE TEXT

ADD THE ARTICLE/BLOG POST TO THE LIST OF DOCUMENTS (CORPUS)

K-MEANS CLUSTERING IN PYTHON

```
THIS BLOCK OF CODE TAKES IN A CORPUS OF DOCUMENTS AND RETURNS A
SET OF VECTORS THAT ARE WEIGHTED BY TF-IDF
vectorizer = TfidfVectorizer(max_df=0.5,min_df=2,stop_words='english')
X = vectorizer.fit_transform(documentCorpus)


HERE WE SPECIFIY THE NUMBER OF CLUSTERS, HOW THE MEANS ARE INITIALIZED
AND THE NUMBER OF ITERATIONS
km = KMeans(n_clusters = 5, init = 'k-means++', max_iter = 100, n_init = 1, verbose = True)
km.fit(X)
```

THESE CLASSES/FUNCTIONS ARE PART OF THE SCI-KIT LEARN PYTHON LIBRARY