```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Dec 18 15:38:19 2019

@author: LeliaSofinezHaouaya
"""
print("Hello World")

#1)Open, import, and/or read a text collection:
import os
DNC_data = '/Users/LeliaSofinezHaouaya/Desktop/UNI <3 /Femte Semester
<3/Open Data Science/Portfolio 2/danish_news_corpus'
DNC_files = []
for file in os.listdir(DNC_data):
    if file.endswith(".txt"):
        filename = os.path.join(DNC_data, file)
        with open(filename, 'r') as document:
            stripped_document_text = ""
            for text in document:
                stripped_document_text = stripped_document_text + " "
+ text.strip()
            DNC_files.append(stripped_document_text)
#Here i have imported the file into python.


#2)Pre-process and describe your collection:
import sklearn
import numpy as np
import pandas as pd
import matplotlib
from pandas import DataFrame
import nltk
#First i need to import important libraries.

nltk.download("stopwords")
from nltk.corpus import stopwords as sw
#I am here downloading the stopwords command.

#Here i am tokenizing documents to count how many words each document
has:
from nltk.tokenize import word_tokenize #I already imported the nltk-
library, and from there on i imported the word_tokenizer.

tokenized_documents = [] #Here i made an empty list called
"tokenized_documents"
for document in DNC_files: #Here i am looping for every document in my
DNC_files which contains my 25 documents from the danish news groups
data-set.
    tokenized_document = word_tokenize(document) #I am bassically
```

```
telling it to word tokenize every word in every 25 document.
    tokenized_documents.append(document) #Here i am appending/filling
out my empty list with the tokenized words.

for tokenized_document in tokenized_documents:
    print(len(tokenized_document))
#This is the total number of words in each of the documents.

all_lenghts = list()
for text in DNC_files:
    all_lenghts.append(len(text))
print("Total sum: %i" % sum(all_lenghts))
#I am making a variable with lenghts of words pr. document. (more
simple then previous command)

#Then i manually picked out the shortest document, and the longest by
using the variables in the beforehand command.
#Shortest document = Index 12 with 2408 characters.
#Longest document = Index 4 with 22266 characters.

from sklearn.feature_extraction.text import CountVectorizer
#Here i imported the CountVectorizer, which can preprocess text, and
also produce document-term matrices.

#3)Select articles using a query:
model_vect = CountVectorizer(stop_words=sw.words('danish'),
token_pattern=r'[a-zA-Z\-][a-zA-Z\-]{2,}')
DNC_vect = model_vect.fit_transform(DNC_files)
#This is my Sparse document term matrix.

import random
random.sample(model_vect.get_feature_names(), 20)
#I am cheking out how the words look.

from sklearn.decomposition import LatentDirichletAllocation
model_lda = LatentDirichletAllocation(n_components=5, random_state=0)
data_lda = model_lda.fit_transform(DNC_vect)
np.shape(data_lda)
#This is topic modeling on my sparse document term matrix.

#4)Model and visualize the topics in your subset:
for i, term_weights in enumerate(model_lda.components_):
    top_idxs = (-term_weights).argsort()[:10]
    top_words = ["%s (%.2f)" % (model_vect.get_feature_names()[idx],
term_weights[idx]) for idx in top_idxs]
    print("Topic %d: %s" % (i, ", ".join(top_words)))
#Here is what my topics contain.

doc_idx = random.randint(0,len(DNC_files)-1)
print('Doc idx: %d' % doc_idx)
```

```
topics = data_lda[doc_idx]
print('Topic vector: %s' % topics)
vote = np.argsort(-topics)[0]
print('Topic vote: %i' % vote)
DNC_files[doc_idx]
#Here i am inspecting a random document (I re-ran this command more
then once to inspect diffrent documents.)

from wordcloud import WordCloud
import matplotlib.pyplot as plt

for i, term_weights in enumerate(model_lda.components_):
    top_idxs = (-term_weights).argsort()[:10]
    top_words = [model_vect.get_feature_names()[idx] for idx in
top_idxs]
    word_freqs = dict(zip(top_words, term_weights[top_idxs]))
    wc = WordCloud(background_color="white",width=300,height=300,
max_words=10).generate_from_frequencies(word_freqs)
    plt.subplot(2, 2, i+1)
    plt.imshow(wc)
#Here i visualized it using WordCloud.

#Done!:D
```