```
# -*- coding: utf-8 -*-
"""Assignment 7 part B.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1j37qB9l2rTWONwttDBxIfqethdFAiI2E
"""

#Algorithm for Create representation of document by calculating TFIDF
#Step 1: Import the necessary libraries
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

#Step 2: Initialize the Documents.
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'

#Step 3: Create BagofWords (BoW) for Document A and B.
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')

#Step 4: Create Collection of Unique words from Document A and B.
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))

#Step 5: Create a dictionary of words and their occurrence for each
document in the corpus
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
  numOfWordsA[word] += 1
  numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
  numOfWordsB[word] += 1

#Step 6: Compute the term frequency for each of our documents.
def computeTF(wordDict, bagOfWords):
  tfDict = {}
  bagOfWordsCount = len(bagOfWords)
  for word, count in wordDict.items():
    tfDict[word] = count / float(bagOfWordsCount)
  return tfDict
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)

#Step 7: Compute the term Inverse Document Frequency.
def computeIDF(documents):
  import math
  N = len(documents)
  idfDict = dict.fromkeys(documents[0].keys(), 0)
  for document in documents:
    for word, val in document.items():
     if val > 0:
       idfDict[word] += 1
  for word, val in idfDict.items():
    idfDict[word] = math.log(N / float(val))
  return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])
```

```
idfs

#Step 8: Compute the term TF/IDF for all words.
def computeTFIDF(tfBagOfWords, idfs):
  tfidf = {}
  for word, val in tfBagOfWords.items():
      tfidf[word] = val * idfs[word]
  return tfidf
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
df = pd.DataFrame([tfidfA, tfidfB])
df
```