2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\rutuja habib\appdata\local\programs\python\python313\lib\site-packages (1.6
Requirement already satisfied: numpy>=1.19.5 in c:\users\rutuja habib\appdata\local\programs\python\python313\lib\site-packages (fro
Requirement already satisfied: scipy>=1.6.0 in c:\users\rutuja habib\appdata\local\programs\python\python313\lib\site-packages (from
Requirement already satisfied: joblib>=1.2.0 in c:\users\rutuja habib\appdata\local\programs\python\python313\lib\site-packages (fro
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\rutuja habib\appdata\local\programs\python\python313\lib\site-packag
```

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import math
```

```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

+ Code + Text

```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1

numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
    numOfWordsB[word] += 1
```

```
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)
```

```
def computeIDF(documents):
    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])
```

```
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], index=['Document A', 'Document B'])
df
```

	planet	Jupiter	Mars	from	fourth	Planet	largest	is	the	Sun
Document A	0.000000	0.138629	0.000000	0.000000	0.000000	0.138629	0.138629	0.0	0.0	0.000000
Document B	0.086643	0.000000	0.086643	0.086643	0.086643	0.000000	0.000000	0.0	0.0	0.086643

Start coding or generate with AI.