Exercise :- 1

```
from sklearn. feature_extraction. text import TfidfVectorizer
  import pandas as pd
  docs =("good movie", "not a good movie", "did not like",
                              "i like it", "good one"
 tfidf = TfidfVectorizer.(min_df=2, max_df=0.5,
                              ngram_range=(1,2))
 features= tfidf.fit transform(docs)
  print (features)
 df =pd. Dataframe(
           features. todense(),
   columns = tfidf . get_features_names())
   print(df)
```

Ex:-2

```
 tfidf = TfidfVectorizer (min_df=1, max_df=0.6,
                                  ngram_range=(1,2))
features = tfidf .fit_transform (docs)
  print(features)
df=pd. Dataframe (features . todense()
  columns = tfidf. get_feature_names ())
   print(df)
```

Ex:-3

```
   from sklearn.metrics.pairwise import
                          linear_kernel.
    doc1 = features [0:1]
    doc2 = features [1:2]
   score =linear_kernel (doc1, doc2)
                          print (score)
```

Natural Language Processing Lab
Lab3. Computing Document Similarity using VSM

EXERCISE-1: Print TFIDF values

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

docs = [
    "good movie", "not a good movie", "did not like",
    "i like it", "good one" ]

# using default tokenizer in TfidfVectorizer
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2))
features = tfidf.fit_transform(docs)
print(features)

# Pretty printing
df = pd.DataFrame(
     features.todense(),
     columns=tfidf.get_feature_names())
print(df)
```

EXERCISE-2:

1. Change the values of **min_df** and **ngram_range** and observe various outputs

EXERCISE-3: Compute Cosine Similarity between 2 Documents

```
from sklearn.metrics.pairwise import linear_kernel

# cosine score between 1st and 2nd doc
doc1 = features[0:1]
doc2 = features[1:2]
score = linear_kernel(doc1, doc2)
print(score)

# cosine score between 1st and all other docs
scores = linear_kernel(doc1, features)
print(scores

# Cosine Similarity for a new doc
query = "I like this good movie"
qfeature = tfidf.transform([query]
scores2 = linear_kernel(doc1, features)
print(scores2)
```

EXERCISE-4: Find Top-N similar documents

Question-1. Consider the following documents and compute TFIDF values

```
docs=["the house had a tiny little mouse",
"the cat saw the mouse",
"the mouse ran away from the house",
"the cat finally ate the mouse",
"the end of the mouse story"
]
```

Question-2. Compute cosine similarity between 3rd document ("`the mouse ran away from the house`") with all other documents. Which is the most similar document?.

Question-3. Find Top-2 similar documents for the 3rd document based on Cosine similarity values.

```
score = linear_kernel(doc1, ~~[illegible]~~ features)
    print(scores)
query = "I like this good movie"
qfeature = tfidf.transform(["query"])
  scor = linear_kernel(doc1, features)
  print(scor)
```

Ex : 4

Quest : 1

```
docs = ["the house had a tiny little mouse",
        "the cat saw the mouse",
        "the mouse ran away from the house",
        "the cat finally ate the mouse",
        "the end of the mouse story"
       ]
```

Ques : 2

```
tfidf = TfidfVectorizer(min_df = 2, max_df = 0.5,
                                   ngram_range = (1,2))
features = tfidf.fit_transform(docs)
print(features)
doc1 = features[0:3]
 s = linear_kernel(doc1, features)
  print(s)
Scores2 = linear_kernel(doc1, features)
  print(scores2)
```