```
# Importing required modules from scikit-learn
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

```
# Sample documents for text similarity comparison
docs = [
    "AI is the future of technology.",
    "Technology and AI are growing fast.",
    "I love reading about machine learning and AI."
]
```

```
# Bag of Words (BoW) Vectorization
# This creates a matrix based on word counts
vectorizer_bow = CountVectorizer()
bow_matrix = vectorizer_bow.fit_transform(docs)

# Calculating cosine similarity on BoW matrix
bow_similarity = cosine_similarity(bow_matrix)
```

```
# Term Frequency (TF) Vectorization
# This is the same as BoW here since CountVectorizer gives raw counts
tf_vectorizer = CountVectorizer()
tf_matrix = tf_vectorizer.fit_transform(docs)

# Calculating cosine similarity on TF matrix
tf_similarity = cosine_similarity(tf_matrix)
```

```
# TF-IDF (Term Frequency - Inverse Document Frequency) Vectorization
# This gives more importance to words that are unique across documents
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(docs)

# Calculating cosine similarity on TF-IDF matrix
tfidf_similarity = cosine_similarity(tfidf_matrix)
```

```
# Printing similarity matrices
print("BoW Similarity:\n", bow_similarity)
print("TF Similarity:\n", tf_similarity)
print("TF-IDF Similarity:\n", tfidf_similarity)
```

```
BoW Similarity:
 [[1.         0.33333333 0.15430335]
 [0.33333333 1.         0.3086067 ]
 [0.15430335 0.3086067  1.        ]]
TF Similarity:
 [[1.         0.33333333 0.15430335]
 [0.33333333 1.         0.3086067 ]
 [0.15430335 0.3086067  1.        ]]
TF-IDF Similarity:
 [[1.         0.19679186 0.06454811]
 [0.19679186 1.         0.17942487]
 [0.06454811 0.17942487 1.        ]]
```