Experiment Number : 6 - Applying similarity measures on the textual datasets and Handle skewness in the dataset.

Batch: FDS-2 **Roll Number: 16010422234** **Name: Chandana Ramesh Galgali**

Aim of the Experiment: Applying similarity measures on the textual datasets and Handle skewness in the dataset.

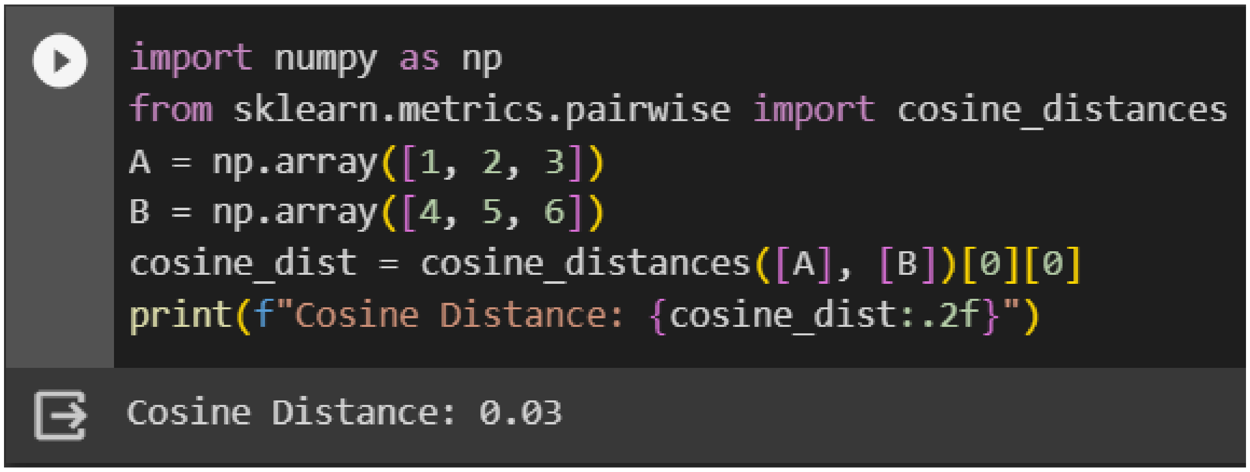
Program/ Steps:

1. Identify the suitable attributes to apply the textual similarity measures among one of them and write python code to calculate Longest common subsequence, edit distance, cosine or Jaccard similarity measures on it.

2. Find skewness in data with any of the correlation coefficient viz. Pearson coefficient, Bowleys coefficient. Finally find the different moments among data. Write a python code for computation of skewness.

Code with Output/Result:

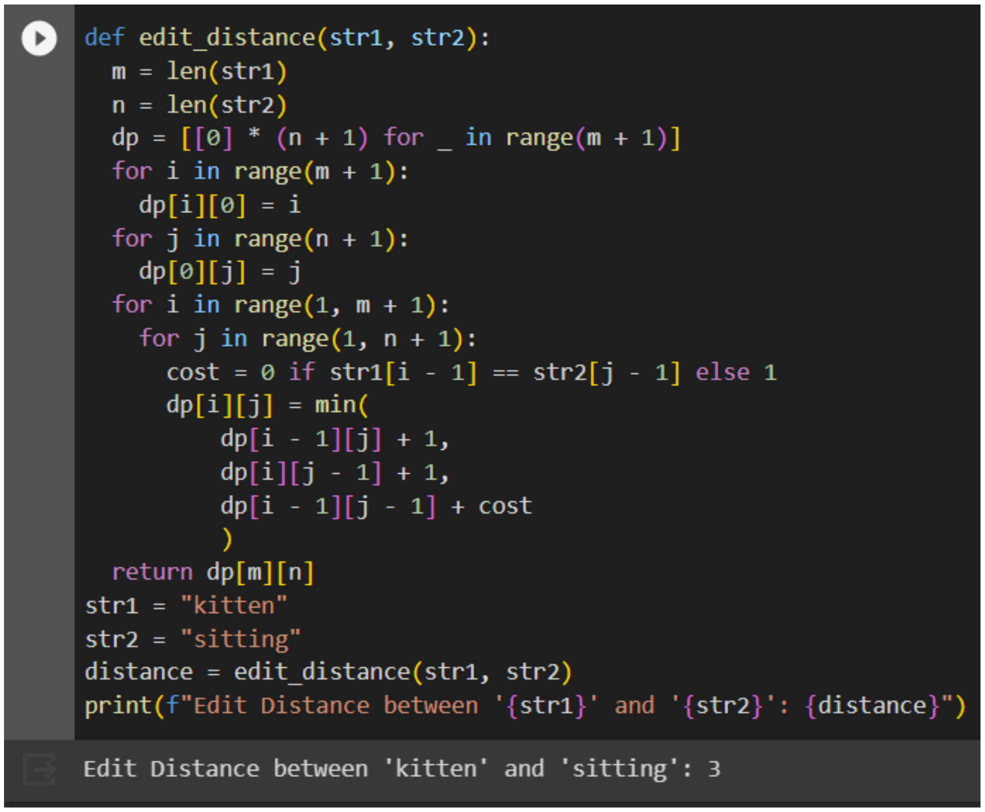
Cosine Distance:



```
import numpy as np
from sklearn.metrics.pairwise import cosine_distances
A = np.array([1, 2, 3])
B = np.array([4, 5, 6])
cosine_dist = cosine_distances([A], [B])[0][0]
print(f"Cosine Distance: {cosine_dist:.2f}")
```

```
Cosine Distance: 0.03
```

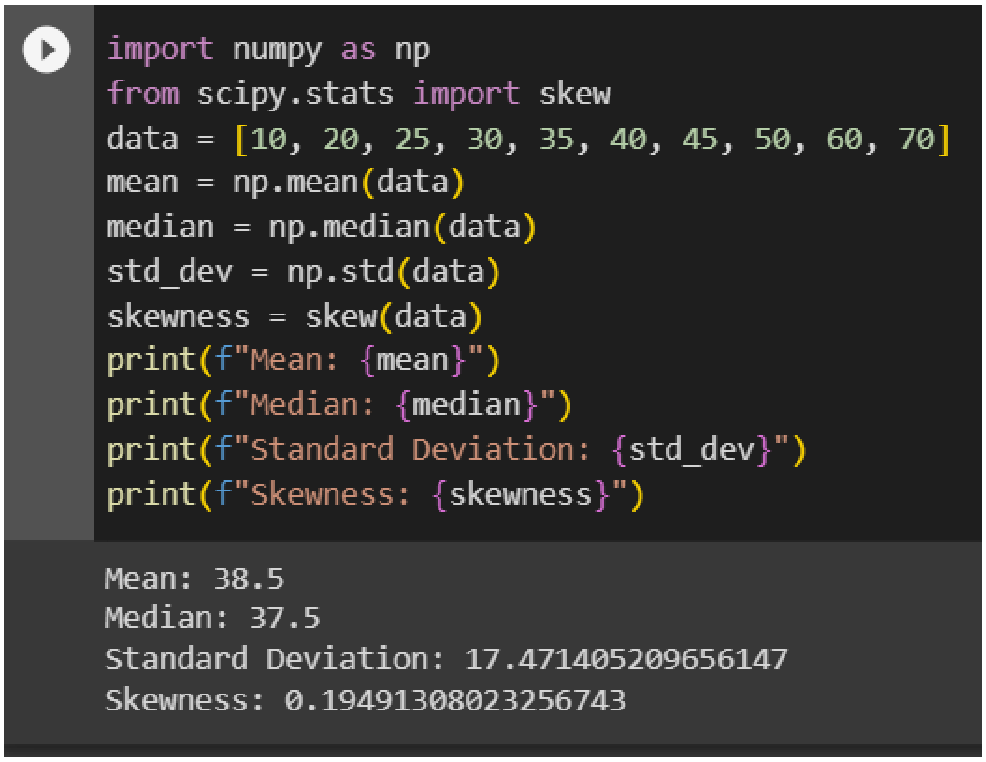
Edit Distance:



```python
def edit_distance(str1, str2):
  m = len(str1)
  n = len(str2)
  dp = [[0] * (n + 1) for _ in range(m + 1)]
  for i in range(m + 1):
    dp[i][0] = i
  for j in range(n + 1):
    dp[0][j] = j
  for i in range(1, m + 1):
    for j in range(1, n + 1):
      cost = 0 if str1[i - 1] == str2[j - 1] else 1
      dp[i][j] = min(
          dp[i - 1][j] + 1,
          dp[i][j - 1] + 1,
          dp[i - 1][j - 1] + cost
          )
  return dp[m][n]
str1 = "kitten"
str2 = "sitting"
distance = edit_distance(str1, str2)
print(f"Edit Distance between '{str1}' and '{str2}': {distance}")
```

```
Edit Distance between 'kitten' and 'sitting': 3
```

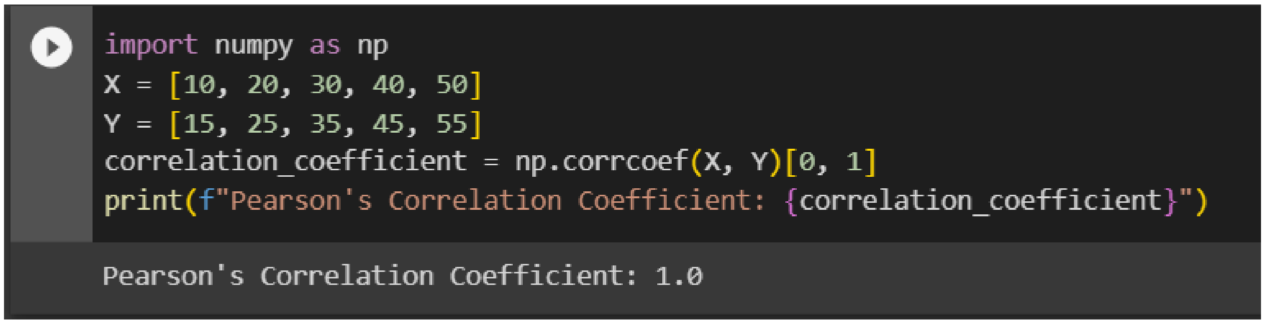
Skewness:



```python
import numpy as np
from scipy.stats import skew
data = [10, 20, 25, 30, 35, 40, 45, 50, 60, 70]
mean = np.mean(data)
median = np.median(data)
std_dev = np.std(data)
skewness = skew(data)
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")
print(f"Skewness: {skewness}")
```

```
Mean: 38.5
Median: 37.5
Standard Deviation: 17.471405209656147
Skewness: 0.19491308023256743
```

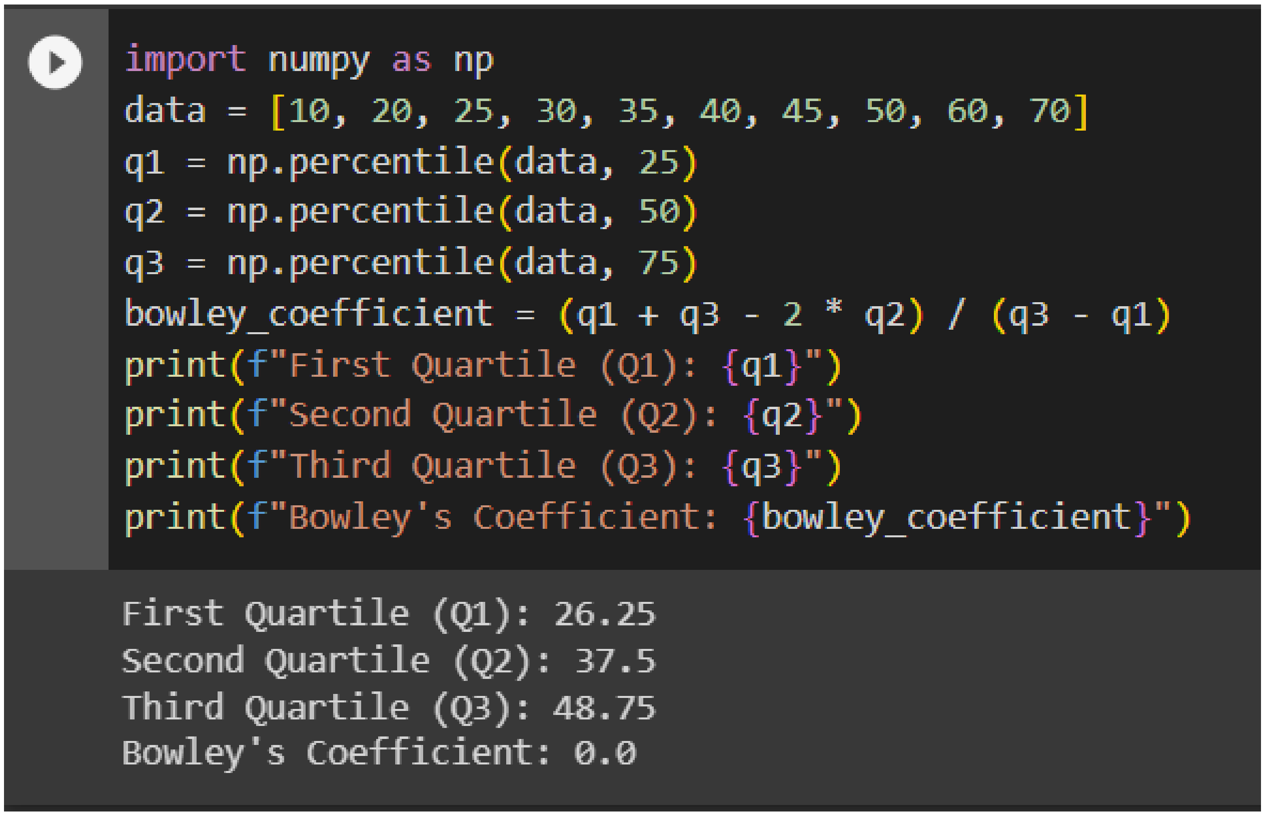
Pearson Coefficient of Correlation:



```
import numpy as np
X = [10, 20, 30, 40, 50]
Y = [15, 25, 35, 45, 55]
correlation_coefficient = np.corrcoef(X, Y)[0, 1]
print(f"Pearson's Correlation Coefficient: {correlation_coefficient}")
```

```
Pearson's Correlation Coefficient: 1.0
```

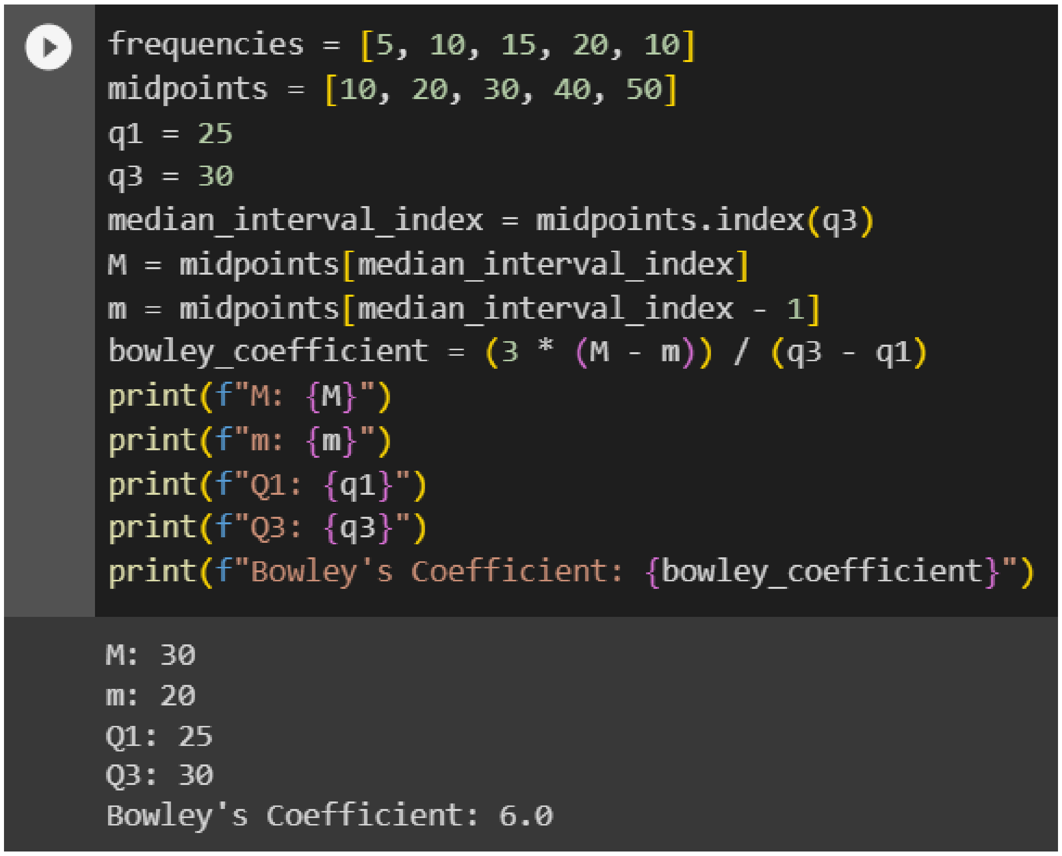
Bowley's skewness coefficient:



```
import numpy as np
data = [10, 20, 25, 30, 35, 40, 45, 50, 60, 70]
q1 = np.percentile(data, 25)
q2 = np.percentile(data, 50)
q3 = np.percentile(data, 75)
bowley_coefficient = (q1 + q3 - 2 * q2) / (q3 - q1)
print(f"First Quartile (Q1): {q1}")
print(f"Second Quartile (Q2): {q2}")
print(f"Third Quartile (Q3): {q3}")
print(f"Bowley's Coefficient: {bowley_coefficient}")
```

```
First Quartile (Q1): 26.25
Second Quartile (Q2): 37.5
Third Quartile (Q3): 48.75
Bowley's Coefficient: 0.0
```

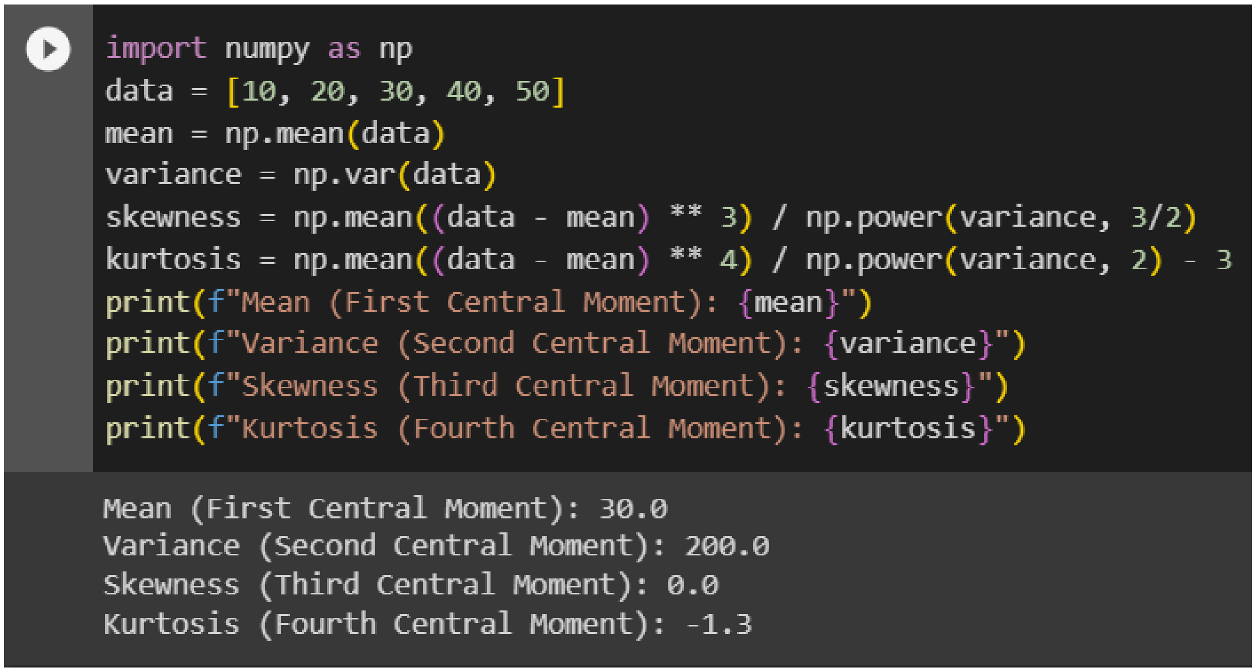
Bowley’s Coefficient for Group Data:



```
frequencies = [5, 10, 15, 20, 10]
midpoints = [10, 20, 30, 40, 50]
q1 = 25
q3 = 30
median_interval_index = midpoints.index(q3)
M = midpoints[median_interval_index]
m = midpoints[median_interval_index - 1]
bowley_coefficient = (3 * (M - m)) / (q3 - q1)
print(f"M: {M}")
print(f"m: {m}")
print(f"Q1: {q1}")
print(f"Q3: {q3}")
print(f"Bowley's Coefficient: {bowley_coefficient}")
```

```
M: 30
m: 20
Q1: 25
Q3: 30
Bowley's Coefficient: 6.0
```

Central moments:



```
import numpy as np
data = [10, 20, 30, 40, 50]
mean = np.mean(data)
variance = np.var(data)
skewness = np.mean((data - mean) ** 3) / np.power(variance, 3/2)
kurtosis = np.mean((data - mean) ** 4) / np.power(variance, 2) - 3
print(f"Mean (First Central Moment): {mean}")
print(f"Variance (Second Central Moment): {variance}")
print(f"Skewness (Third Central Moment): {skewness}")
print(f"Kurtosis (Fourth Central Moment): {kurtosis}")
```

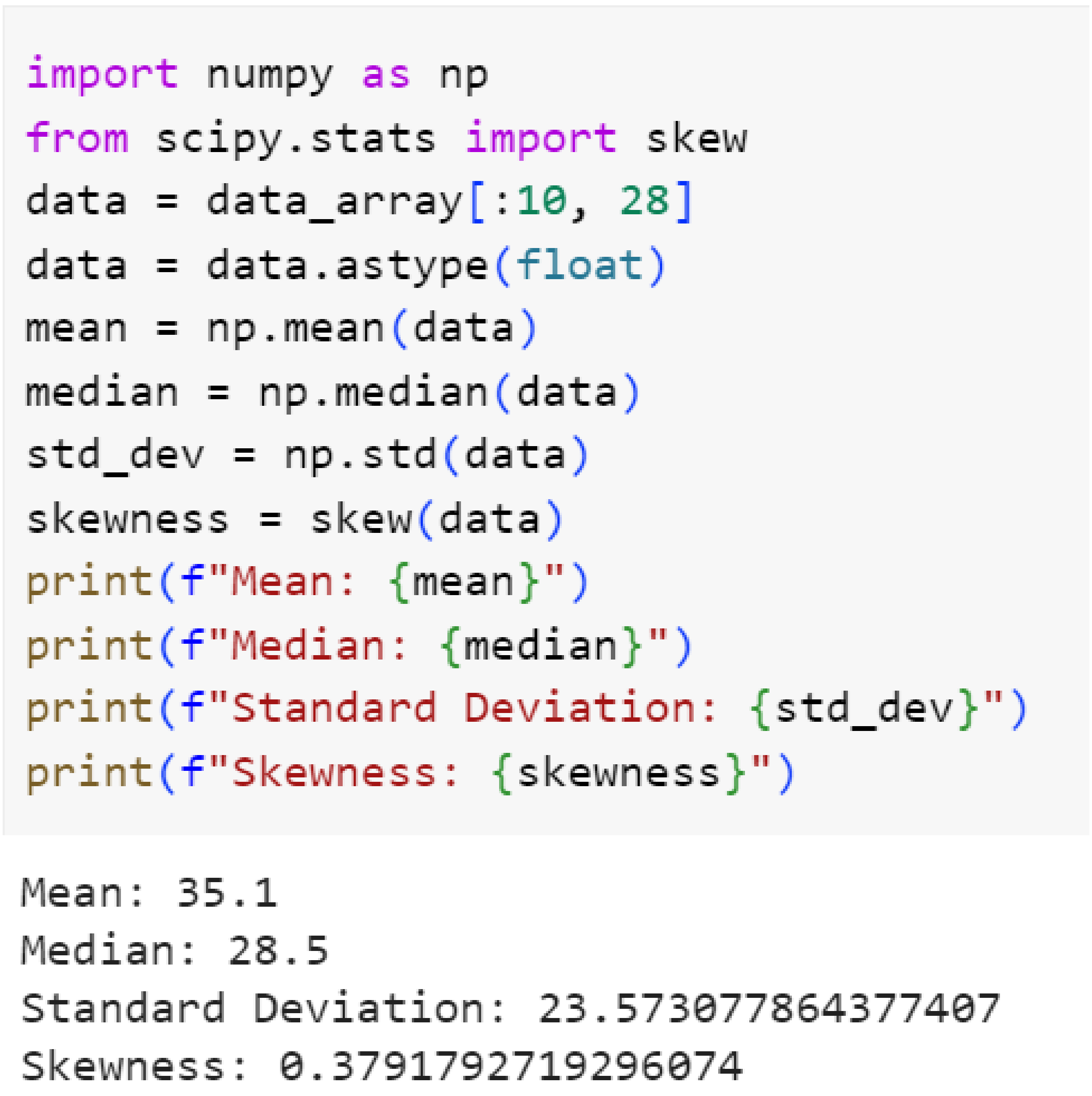
```
Mean (First Central Moment): 30.0
Variance (Second Central Moment): 200.0
Skewness (Third Central Moment): 0.0
Kurtosis (Fourth Central Moment): -1.3
```

```
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_distances
data = pd.read_csv(r'/Flight_delay.csv')
data_array = data.to_numpy()
Arrival_Delay = data_array[:5, 12]
Departure_Delay = data_array[:5, 13]
cosine_dist = cosine_distances([Arrival_Delay], [Departure_Delay])[0][0]
print(f"Cosine Distance between Arrival Delay and Departure Delay: {cosine_dist:.2f}")
```

```
Cosine Distance between Arrival Delay and Departure Delay: 0.01
```

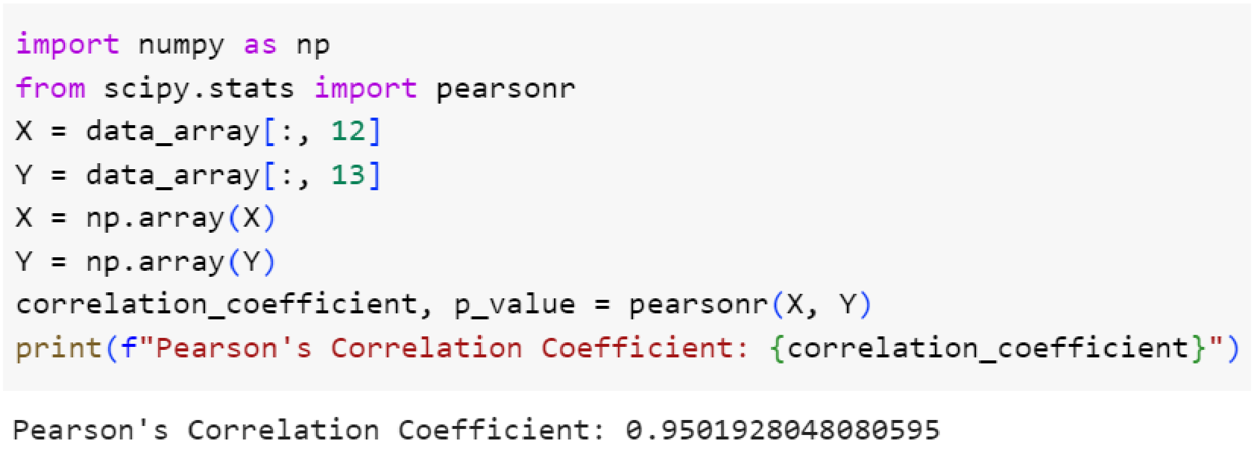
```
def edit_distance(str1, str2):
  m = len(str1)
  n = len(str2)
  dp = [[0] * (n + 1) for _ in range(m + 1)]
  for i in range(m + 1):
    dp[i][0] = i
  for j in range(n + 1):
    dp[0][j] = j
  for i in range(1, m + 1):
    for j in range(1, n + 1):
      cost = 0 if str1[i - 1] == str2[j - 1] else 1
      dp[i][j] = min(
          dp[i - 1][j] + 1,
          dp[i][j - 1] + 1,
          dp[i - 1][j - 1] + cost
          )
  return dp[m][n]
str1 = data_array[1:2, 6]
str2 = data_array[17021:17022, 6]
distance = edit_distance(str1, str2)
print(f"Edit Distance between '{str1}' and '{str2}': {distance}")
```

```
Edit Distance between '['Southwest Airlines Co.']' and '['Skywest Airlines Inc.']': 1
```



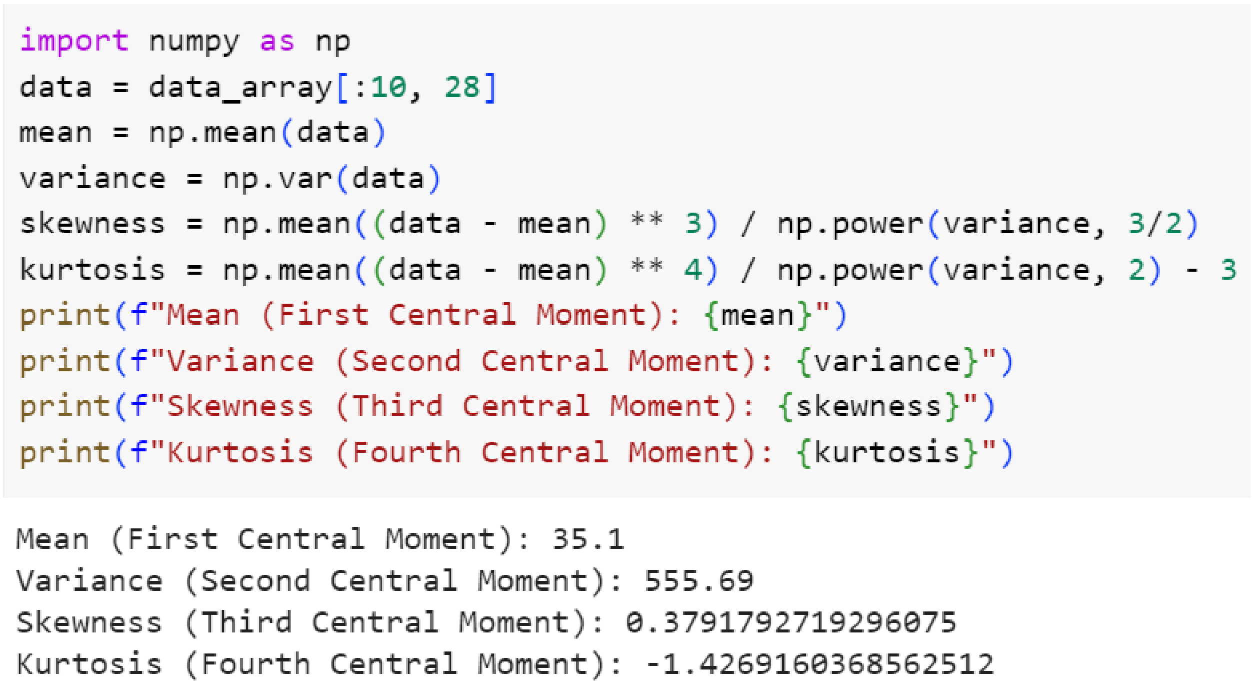
```python
import numpy as np
from scipy.stats import skew
data = data_array[:10, 28]
data = data.astype(float)
mean = np.mean(data)
median = np.median(data)
std_dev = np.std(data)
skewness = skew(data)
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")
print(f"Skewness: {skewness}")
```

```
Mean: 35.1
Median: 28.5
Standard Deviation: 23.573077864377407
Skewness: 0.379179271929 6074
```



```
import numpy as np
from scipy.stats import pearsonr
X = data_array[:, 12]
Y = data_array[:, 13]
X = np.array(X)
Y = np.array(Y)
correlation_coefficient, p_value = pearsonr(X, Y)
print(f"Pearson's Correlation Coefficient: {correlation_coefficient}")
```

```
Pearson's Correlation Coefficient: 0.9501928048080595
```



```
import numpy as np
data = data_array[:10, 28]
mean = np.mean(data)
variance = np.var(data)
skewness = np.mean((data - mean) ** 3) / np.power(variance, 3/2)
kurtosis = np.mean((data - mean) ** 4) / np.power(variance, 2) - 3
print(f"Mean (First Central Moment): {mean}")
print(f"Variance (Second Central Moment): {variance}")
print(f"Skewness (Third Central Moment): {skewness}")
print(f"Kurtosis (Fourth Central Moment): {kurtosis}")
```

```
Mean (First Central Moment): 35.1
Variance (Second Central Moment): 555.69
Skewness (Third Central Moment): 0.3791792719296075
Kurtosis (Fourth Central Moment): -1.4269160368562512
```

Post Lab Question-Answers:

1. What are the different applications of Textual similarity measure?

Ans: Textual similarity measures have various applications across different domains. Some of the common applications include:

1. Information Retrieval: Textual similarity measures are used in search engines to retrieve relevant documents based on the similarity between the query and the documents in the database.

2. Plagiarism Detection: Textual similarity measures can be used to identify instances of plagiarism by comparing a given document with a set of reference documents.

3. Question Answering Systems: Textual similarity measures are employed in question answering systems to match user queries with relevant answers or documents.

4. Document Clustering: Textual similarity measures can be used to group similar documents together in document clustering tasks, which can aid in organizing large document collections.

5. Recommender Systems: Textual similarity measures can be utilized in recommender systems to recommend similar items or content based on the similarity between user preferences and item descriptions.

6. Sentiment Analysis: Textual similarity measures can be applied in sentiment analysis tasks to compare the similarity between different text snippets or to identify similar sentiment expressions.

7. Machine Translation: Textual similarity measures can assist in machine translation tasks by comparing the similarity between source and target language sentences to improve translation quality.

8. Text Summarization: Textual similarity measures can be used in text summarization tasks to identify similar sentences or passages and generate concise summaries.

9. Natural Language Processing (NLP): Textual similarity measures are widely used in various NLP applications, such as text classification, named entity recognition, and information extraction.

These are just a few examples of the applications of textual similarity measures, and they can be applied in many other areas where text analysis and comparison are required.

2. What are the different applications of finding similarity between textual attributes?

Ans: Finding similarity between textual attributes has several applications in various domains. Some of the common applications include:

1. Product Recommendations: Similarity between textual attributes of products can be used to recommend similar products to customers. For example, in e-commerce, if a customer is viewing a particular product, the system can recommend similar products based on the similarity of their attributes.

2. Content Filtering: Similarity between textual attributes can be used in content filtering systems to identify and filter out duplicate or near-duplicate content. This is useful in preventing redundancy and maintaining content quality.

3. Document Clustering: Similarity between textual attributes can be used to cluster documents with similar content together. This can aid in organizing large document collections and facilitating efficient retrieval and browsing.

4. Entity Matching: Similarity between textual attributes can be used in entity matching tasks to identify and match entities across different datasets. This is useful in data integration and data cleaning processes.

5. Text Categorization: Similarity between textual attributes can be used in text categorization tasks to classify documents into predefined categories. By comparing the similarity of the document's attributes with the attributes of known categories, the system can assign the document to the most similar category.

6. Information Retrieval: Similarity between textual attributes can be used in search engines to retrieve relevant documents based on the similarity between the query and the attributes of the documents in the database.

7. Fraud Detection: Similarity between textual attributes can be used in fraud detection systems to identify patterns of fraudulent behavior. By comparing the similarity of attributes across different transactions or user profiles, suspicious activities can be detected.

8. Sentiment Analysis: Similarity between textual attributes can be used in sentiment analysis tasks to compare the similarity between different text snippets or to identify similar sentiment expressions.

These are just a few examples of the applications of finding similarity between textual attributes. The ability to measure similarity between textual attributes is valuable in many other areas where text analysis and comparison are required.

Outcomes: Comprehend descriptive and proximity measures of data

---

Conclusion (based on the Results and outcomes achieved):

This experiment highlights the importance of applying similarity measures on textual datasets and the need to handle skewness for reliable and effective data analysis. The findings contribute to the advancement of text mining, information retrieval, and other related fields, paving the way for improved decision-making processes and enhanced user experiences.

References:

Books/ Journals/ Websites

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3nd Edition

---