In [7]:
```python
import re

def replace_space_comma_dot(input_string):
    pattern = re.compile(r'[ ,.]')
    replaced_string = re.sub(pattern, ':', input_string)
    return replaced_string

# Test the function
sample_text = 'India is the best country of the world, with some dots. Let\'s replace them.'
result = replace_space_comma_dot(sample_text)
print(result)
```
```
This:is:a:test::with:some:dots::Let's:replace:them:
```

In [9]:
```python
import re

def replace_space_comma_dot(input_string):
    pattern = re.compile(r'[ ,.]')
    replaced_string = re.sub(pattern, ':', input_string)
    return replaced_string

# Test the function
sample_text = 'India is the best country of the world, with some dots. Let\'s replace them.'
result = replace_space_comma_dot(sample_text)
print(result)
```
```
India:is:the:best:country:of:the:world::with:some:dots::Let's:replace:them:
```

In [10]:
```python
import pandas as pd
import re

# Define the dictionary
data = {'SUMMARY': ['hello, python!', 'XXXXX test', '123four, five:; six...']}

# Create a dataframe
df = pd.DataFrame(data)

# Function to remove unwanted characters
def remove_unwanted_chars(text):
    cleaned_text = re.sub(r'[^a-zA-Z\s]', '', text)
    return cleaned_text.strip()

# Apply the function to the 'SUMMARY' column
df['SUMMARY'] = df['SUMMARY'].apply(remove_unwanted_chars)

# Display the resulting dataframe
print(df)
```
```
         SUMMARY
0    hello world
1     XXXXX test
2  four five six
```

In [11]:
```python
def find_words_at_least_four_chars(input_string):
    pattern = re.compile(r'\b\w{4,}\b')
    matches = pattern.findall(input_string)
    return matches

# Test the function
sample_text = 'This is a test sentence with some words of varying lengths.'
result = find_words_at_least_four_chars(sample_text)
print(result)
```
```
['This', 'test', 'sentence', 'with', 'some', 'words', 'varying', 'lengths']
```

In [12]:
```python
def find_words_of_length(input_string, min_length=3, max_length=5):
    pattern = re.compile(r'\b\w{%d,%d}\b' % (min_length, max_length))
    matches = pattern.findall(input_string)
    return matches

# Test the function
sample_text = 'This is a test sentence with some words of varying lengths.'
result = find_words_of_length(sample_text, 3, 5)
print(result)
```
```
['This', 'test', 'with', 'some', 'words']
```

In [13]:
```python
import re

def remove_parentheses(strings_list):
    pattern = re.compile(r'\([^)]*\)')
    cleaned_strings = [pattern.sub('', s) for s in strings_list]
    return cleaned_strings

# Test the function
sample_text = ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data Science World)", "Data (Scientist)"]
result = remove_parentheses(sample_text)
print(result)
```
```
['example ', 'hr@fliprobo ', 'github ', 'Hello ', 'Data ']
```

In [14]:
```python
import re

# Step 1: Save the sample text to a text file
sample_text = ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data Science World)", "Data (Scientist)"]
file_path = 'sample_text.txt'

with open(file_path, 'w') as file:
    for line in sample_text:
        file.write(line + '\n')

# Step 2: Read the text from the file
with open(file_path, 'r') as file:
    text = file.read()

# Step 3: Use a regular expression to remove the parenthesis area
pattern = re.compile(r'\([^)]*\)')
cleaned_text = pattern.sub('', text)

# Step 4: Write the modified text back to the file
with open(file_path, 'w') as file:
    file.write(cleaned_text)

print(f"Original text:\n{text}\n")
print(f"Modified text:\n{cleaned_text}")
```
```
Original text:
example (.com)
hr@fliprobo (.com)
github (.com)
Hello (Data Science World)
Data (Scientist)


Modified text:
example
hr@fliprobo
github
Hello
Data
```

In [15]:
```python
import re

sample_text = "ImportanceOfRegularExpressionsInPython"

# Use regular expression to split string into uppercase letters
result = re.findall(r'[A-Z][a-z]*', sample_text)

print(result)
```
```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

In [16]:
```python
import re

def insert_spaces(text):
    # Use regular expression to insert spaces between words starting with numbers
    result = re.sub(r'(\d)([A-Za-z])', r'\1 \2', text)
    return result

sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Call the function
output_text = insert_spaces(sample_text)

print("Original text:", sample_text)
print("Modified text:", output_text)
```
```
Original text: RegularExpression1IsAn2ImportantTopic3InPython
Modified text: RegularExpression1 IsAn2 ImportantTopic3 InPython
```

In [17]:
```python
import re

def insert_spaces(text):
    # Use regular expression to insert spaces between words starting with capital letters or numbers
    result = re.sub(r'([A-Z\d])([a-z])', r'\1 \2', text)
    return result

sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Call the function
output_text = insert_spaces(sample_text)

print("Original text:", sample_text)
print("Modified text:", output_text)
```
```
Original text: RegularExpression1IsAn2ImportantTopic3InPython
Modified text: R egularE xpression1I sA n2I mportantT opic3I nP ython
```

In [18]:
```python
import re

def match_string(input_string):
    # Define the regular expression pattern
    pattern = re.compile(r'^[a-zA-Z0-9_]+$')

    # Use the pattern to match the input string
    match_result = re.match(pattern, input_string)

    if match_result:
        return True
    else:
        return False

# Test the function with different strings
test_string1 = "Hello_World123"
test_string2 = "Special@Characters"
test_string3 = "123_numbers"

print(f"{test_string1}: {match_string(test_string1)}")
print(f"{test_string2}: {match_string(test_string2)}")
print(f"{test_string3}: {match_string(test_string3)}")
```
```
Hello_World123: True
Special@Characters: False
123_numbers: True
```

In [19]:
```python
def starts_with_number(input_string, specific_number):
    # Check if the input string starts with the specific number
    return input_string.startswith(str(specific_number))

# Test the function with different strings and a specific number
test_string1 = "123abc"
test_string2 = "456xyz"
specific_number = 123

print(f"{test_string1} starts with {specific_number}: {starts_with_number(test_string1, specific_number)}")
print(f"{test_string2} starts with {specific_number}: {starts_with_number(test_string2, specific_number)}")
```
```
123abc starts with 123: True
456xyz starts with 123: False
```

In [20]:
```python
def remove_leading_zeros(ip_address):
    # Split the IP address into octets
    octets = ip_address.split('.')

    # Remove leading zeros from each octet
    cleaned_octets = [str(int(octet)) for octet in octets]

    # Join the octets back into an IP address
    cleaned_ip_address = '.'.join(cleaned_octets)

    return cleaned_ip_address

# Test the function with an example IP address
example_ip_address = "192.168.001.001"
cleaned_ip_address = remove_leading_zeros(example_ip_address)

print(f"Original IP address: {example_ip_address}")
print(f"Cleaned IP address: {cleaned_ip_address}")
```
```
Original IP address: 192.168.001.001
Cleaned IP address: 192.168.1.1
```

In [21]:
```python
import re

def extract_dates_from_text_file(file_path):
    with open(file_path, 'r') as file:
        text = file.read()

    # Define the regular expression pattern for extracting date strings
    date_pattern = r'\b(?:January|February|March|April|May|June|July|August|September|October|November|December)\s+\d{1,2}(?:st|nd|rd|th)\s+\d{4}\b'

    # Find all matches in the text
    date_matches = re.findall(date_pattern, text)

    return date_matches

# Test the function with the given sample text file
file_path = 'sample_text.txt'  # Replace with the actual path to your text file
dates = extract_dates_from_text_file(file_path)

print("Extracted Date Strings:")
for date in dates:
    print(date)
```
```
Extracted Date Strings:
```

In [22]:
```python
def search_literals_in_text(text, searched_words):
    found_words = [word for word in searched_words if word in text]
    return found_words

# Sample text
sample_text = 'The quick brown fox jumps over the lazy dog.'

# Words to search for
searched_words = ['fox', 'dog', 'horse']

# Search for the specified words in the text
found_words = search_literals_in_text(sample_text, searched_words)

# Print the results
print("Words found in the text:")
for word in found_words:
    print(word)
```
```
Words found in the text:
fox
dog
```

In [23]:
```python
def search_literal_in_text(text, searched_word):
    locations = []
    start = text.find(searched_word)

    while start != -1:
        locations.append(start)
        start = text.find(searched_word, start + 1)

    return locations

# Sample text
sample_text = 'The quick brown fox jumps over the lazy dog.'

# Word to search for
searched_word = 'fox'

# Search for the specified word in the text
found_locations = search_literal_in_text(sample_text, searched_word)

# Print the results
if found_locations:
    print(f"The word '{searched_word}' was found at the following location(s): {found_locations}")
else:
    print(f"The word '{searched_word}' was not found in the text.")
```
```
The word 'fox' was found at the following location(s): [16]
```

In [24]:
```python
import re

def find_substrings(text, pattern):
    substrings = re.findall(pattern, text)
    return substrings

# Sample text
sample_text = 'Python exercises, PHP exercises, C# exercises'

# Pattern to search for
search_pattern = 'exercises'

# Find all substrings based on the pattern
found_substrings = find_substrings(sample_text, search_pattern)

# Print the results
if found_substrings:
    print(f"The substrings matching the pattern '{search_pattern}' are: {found_substrings}")
else:
    print(f"No substrings matching the pattern '{search_pattern}' were found in the text.")
```
```
The substrings matching the pattern 'exercises' are: ['exercises', 'exercises', 'exercises']
```

In [25]:
```python
def find_occurrences_positions(text, substring):
    occurrences = []
    start = 0

    while start < len(text):
        position = text.find(substring, start)
        if position == -1:
            break
        occurrences.append((position, position + len(substring) - 1))
        start = position + 1

    return occurrences

# Sample text
sample_text = 'Python exercises, PHP exercises, C# exercises'

# Substring to search for
search_substring = 'exercises'

# Find occurrences and positions of the substring
found_occurrences = find_occurrences_positions(sample_text, search_substring)

# Print the results
if found_occurrences:
    print(f"The substring '{search_substring}' occurs at the following positions:")
    for start, end in found_occurrences:
        print(f"Position: {start} - {end}")
else:
    print(f"No occurrences of the substring '{search_substring}' were found in the text.")
```
```
The substring 'exercises' occurs at the following positions:
Position: 7 - 15
Position: 22 - 30
Position: 36 - 44
```

In [26]:
```python
from datetime import datetime

def convert_date_format(input_date):
    try:
        # Parse the input date
        parsed_date = datetime.strptime(input_date, '%Y-%m-%d')

        # Convert the date to the desired format
        formatted_date = parsed_date.strftime('%d-%m-%Y')

        return formatted_date
    except ValueError:
        return "Invalid date format. Please use 'yyyy-mm-dd'."

# Example usage
input_date = '2023-11-25'
converted_date = convert_date_format(input_date)

print(f"Original date: {input_date}")
print(f"Converted date: {converted_date}")
```
```
Original date: 2023-11-25
Converted date: 25-11-2023
```

In [27]:
```python
import re

def find_decimal_numbers(input_text):
    try:
        # Define the regular expression pattern
        pattern = re.compile(r'\b\d+\.\d{1,2}\b')

        # Find all matches in the input text
        matches = re.findall(pattern, input_text)

        return matches
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"
decimal_numbers = find_decimal_numbers(sample_text)

print(f"Sample Text: {sample_text}")
print(f"Decimal Numbers with Precision 1 or 2: {decimal_numbers}")
```
```
Sample Text: 01.12 0132.123 2.31875 145.8 3.01 27.25 0.25
Decimal Numbers with Precision 1 or 2: ['01.12', '145.8', '3.01', '27.25', '0.25']
```

In [28]:
```python
import re

def separate_numbers_and_positions(input_text):
    try:
        # Define the regular expression pattern to find numbers
        number_pattern = re.compile(r'\b\d+\b')

        # Find all matches of numbers in the input text
        numbers = re.findall(number_pattern, input_text)

        # Get the positions of the numbers
        positions = [match.start() for match in re.finditer(number_pattern, input_text)]

        # Combine numbers and their positions
        result = list(zip(numbers, positions))

        return result
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_text = "The price of the product is $20.99 and the quantity is 15."
result = separate_numbers_and_positions(sample_text)

print(f"Sample Text: {sample_text}")
print(f"Numbers and Their Positions: {result}")
```
```
Sample Text: The price of the product is $20.99 and the quantity is 15.
Numbers and Their Positions: [('20', 29), ('99', 32), ('15', 55)]
```

In [29]:
```python
import re

def extract_maximum_numeric_value(input_text):
    try:
        # Define the regular expression pattern to find numbers
        number_pattern = re.compile(r'\b\d+\b')

        # Find all matches of numbers in the input text
        numbers = list(map(int, re.findall(number_pattern, input_text)))

        # Extract the maximum numeric value
        max_value = max(numbers)

        return max_value
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_text = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'
result = extract_maximum_numeric_value(sample_text)

print(f"Sample Text: {sample_text}")
print(f"Maximum Numeric Value: {result}")
```
```
Sample Text: My marks in each semester are: 947, 896, 926, 524, 734, 950, 642
Maximum Numeric Value: 950
```

In [30]:
```python
import re

def insert_spaces_before_capital(text):
    try:
        # Define the regular expression pattern to find words starting with capital letters
        pattern = re.compile(r'(?<=[a-z])([A-Z])')

        # Insert spaces before capital letters
        spaced_text = re.sub(pattern, r' \1', text)

        return spaced_text
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_text = "RegularExpressionIsAnImportantTopicInPython"
result = insert_spaces_before_capital(sample_text)

print(f"Original Text: {sample_text}")
print(f"Modified Text: {result}")
```
```
Original Text: RegularExpressionIsAnImportantTopicInPython
Modified Text: Regular Expression Is An Important Topic In Python
```

In [31]:
```python
import re

def find_sequences(text):
    try:
        # Define the regular expression pattern
        pattern = re.compile(r'\b[A-Z][a-z]*\b')

        # Find sequences
        sequences = re.findall(pattern, text)

        return sequences
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_text = "The Quick Brown Fox Jumps Over The Lazy Dog"
result = find_sequences(sample_text)

print(f"Original Text: {sample_text}")
print(f"Found Sequences: {result}")
```
```
Original Text: The Quick Brown Fox Jumps Over The Lazy Dog
Found Sequences: ['The', 'Quick', 'Brown', 'Fox', 'Jumps', 'Over', 'The', 'Lazy', 'Dog']
```

In [32]:
```python
import re

def ends_with_alphanumeric(input_string):
    try:
        # Define the regular expression pattern
        pattern = re.compile(r'.*\w$')

        # Check if the input string ends with an alphanumeric character
        match = re.match(pattern, input_string)

        if match:
            return True
        else:
            return False
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_string1 = "Hello123"
sample_string2 = "Greeting!"
sample_string3 = "12345"

print(f"{sample_string1} ends with alphanumeric: {ends_with_alphanumeric(sample_string1)}")
print(f"{sample_string2} ends with alphanumeric: {ends_with_alphanumeric(sample_string2)}")
print(f"{sample_string3} ends with alphanumeric: {ends_with_alphanumeric(sample_string3)}")
```
```
Hello123 ends with alphanumeric: True
Greeting! ends with alphanumeric: False
12345 ends with alphanumeric: True
```

In [33]:
```python
import re

def extract_hashtags(sample_text):
    try:
        # Define the regular expression pattern for extracting hashtags
        pattern = re.compile(r'#\w+')

        # Use findall to extract all occurrences of the pattern in the text
        hashtags = re.findall(pattern, sample_text)

        return hashtags
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same
                has rendered USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo"""

hashtags_list = extract_hashtags(sample_text)
print("Extracted Hashtags:", hashtags_list)
```
```
Extracted Hashtags: ['#Doltiwal', '#xyzabc', '#Demonetization']
```

In [34]:
```python
import re

def remove_special_symbols(sample_text):
    try:
        # Define the regular expression pattern for <U+..> symbols
        pattern = re.compile(r'<U\+\w+>')

        # Use sub to replace all occurrences of the pattern with an empty string
        cleaned_text = re.sub(pattern, '', sample_text)

        return cleaned_text
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_text = "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are protesting #demonetization are all different party leaders"

cleaned_text = remove_special_symbols(sample_text)
print("Cleaned Text:", cleaned_text)
```
```
Cleaned Text: @Jags123456 Bharat band on 28??<ed><ed>Those who are protesting #demonetization are all different party leaders
```

In [35]:
```python
import re

def extract_dates_from_file(file_path):
    try:
        # Read the content of the file
        with open(file_path, 'r') as file:
            text = file.read()

        # Define the regular expression pattern for dates in the format DD-MM-YYYY
        pattern = re.compile(r'\b\d{2}-\d{2}-\d{4}\b')

        # Use findall to extract all occurrences of the pattern
        dates = re.findall(pattern, text)

        return dates
    except Exception as e:
        return f"Error: {e}"

# Example usage
file_path = 'sample_text.txt'  # Replace with the actual path to your text file
extracted_dates = extract_dates_from_file(file_path)

print("Extracted Dates:", extracted_dates)
```
```
Extracted Dates: []
```

In [36]:
```python
import re

def remove_words_of_length_between_2_and_4(input_string):
    try:
        # Define the regular expression pattern for words of length 2 to 4
        pattern = re.compile(r'\b\w{2,4}\b')

        # Use sub to replace all occurrences of the pattern with an empty string
        result_string = re.sub(pattern, '', input_string)

        return result_string.strip()  # Remove leading and trailing spaces
    except Exception as e:
        return f"Error: {e}"

# Example usage
sample_text = "The following example creates an ArrayList with a capacity of 50 elements. 4 elements are then added to the ArrayList and the ArrayList is trimmed accordingly."
result = remove_words_of_length_between_2_and_4(sample_text)

print("Result:", result)
```
```
Result: following example creates  ArrayList  a capacity   elements. 4 elements   added   ArrayList   ArrayList  trimmed accordingly.
```

In [ ]: