In [1]:
```
# Question 1- Write a Python program to replace all occurrences of a space, comma,
#Sample Text- 'Python Exercises, PHP exercises.'
# Expected Output: Python:Exercises::PHP:exercises:
# Answer:
def replace_characters(text):

    characters_to_replace = [' ', ',', '.']

    for char in characters_to_replace:
        text = text.replace(char, ':')

    return text


sample_text = 'Python Exercises, PHP exercises.'

result = replace_characters(sample_text)
print(result)
```

```
Python:Exercises::PHP:exercises:
```

In [56]:
```
#Question 2-  Create a dataframe using the dictionary below and remove everything (
#from the columns except words.
#- {'SUMMARY' : ['hello, world!', 'XXXXX test', '123four, five:; six...']}
#Expected output-
# 0      hello world
#  1             test
#  2    four five six
# Answer:
import pandas as pd
import re

# Dictionary
data = {'SUMMARY': ['hello, world!', 'XXXXX test', '123four, five:; six...']}

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

# Function to clean text
def clean_text(text):
    # Remove everything except words
    cleaned_text = re.sub(r'[^a-zA-Z\s]', '', text)
    return cleaned_text.strip()

# Apply the clean_text function to each cell in the 'SUMMARY' column
df['SUMMARY'] = df['SUMMARY'].apply(clean_text)

# Display the result
print(df)
```

```
         SUMMARY
0    hello world
1     XXXXX test
2  four five six
```

In [3]:
```
# Question 3- Create a function in python to find all words that are at least 4 cha
#The use of the re.compile() method is mandatory.
#Answer:
import re

def find_long_words(input_string):
    # Define a regular expression pattern to match words of at least 4 characters
    pattern = re.compile(r'\b\w{4,}\b')
```

```
    # Use findall to get all matches in the input string
    long_words = pattern.findall(input_string)

    return long_words

# Example usage
input_text = "My Name Is Madhuri"

result = find_long_words(input_text)
print(result)
```

```
['Name', 'Madhuri']
```

In [4]:
```
# Question 4- Create a function in python to find all three, four, and five charact
# The use of the re.compile() method is mandatory.
import re

def find_words_by_length(input_string):
    # Define a regular expression pattern to match words of three, four, or five ch
    pattern = re.compile(r'\b\w{3,5}\b')

    # Use findall to get all matches in the input string
    matching_words = pattern.findall(input_string)

    return matching_words

# Example usage
input_text = "This is a sample sentence with various words of different lengths."

result = find_words_by_length(input_text)
print(result)
```

```
['This', 'with', 'words']
```

In [8]:
```
#Question 5- Create a function in Python to remove the parenthesis in a list of str
#The use of the re.compile() method is mandatory.
#Sample Text: ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Dat
#Expected Output:
#example.com
#hr@fliprobo.com
#github.com
#Hello Data Science World
#Data Scientist
import re

def remove_parentheses(strings_list):
    # Define a regular expression pattern to match parentheses and their contents
    pattern = re.compile(r'\([^)]*\)')

    # Use sub to replace matches with an empty string for each string in the list
    cleaned_strings = [pattern.sub('', string).strip() for string in strings_list]

    return cleaned_strings

# Sample text
sample_text = ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Dat

# Apply the function and print the result
result = remove_parentheses(sample_text)
for cleaned_string in result:
    print(cleaned_string)
```

```
example
hr@fliprobo
github
Hello
Data
```

In [22]:
```python
#Question 7- Write a regular expression in Python to split a string into uppercase
#Sample text: “ImportanceOfRegularExpressionsInPython”
#Expected Output: [‘Importance’, ‘Of’, ‘Regular’, ‘Expression’, ‘In’, ‘Python’]
#Answer:
import re

sample_text = "ImportanceOfRegularExpressionsInPython"

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

print(result)
```

```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

In [23]:
```python
#Question 8- Create a function in python to insert spaces between words starting wi
#Sample Text: “RegularExpression1IsAn2ImportantTopic3InPython"
#Expected Output: RegularExpression 1IsAn 2ImportantTopic 3InPython
#Answer:
import re

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

# Sample text
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Call the function
result = insert_spaces(sample_text)

# Display the result
print(result)
```

```
RegularExpression1 IsAn2 ImportantTopic3 InPython
```

In [31]:
```python
#Question 9- Create a function in python to insert spaces between words starting wi
#Sample Text: “RegularExpression1IsAn2ImportantTopic3InPython"
#Expected Output:  RegularExpression 1 IsAn 2 ImportantTopic 3 InPython
import re

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

# Sample text
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Call the function
result = insert_spaces(sample_text)

# Display the result
print(result)
```

```
RegularExpression1 IsAn2 ImportantTopic3 InPython
```

In [32]:
```
#Question 11- Write a Python program to match a string that contains only upper and

import re

def is_valid_string(input_str):
    # Define the regular expression pattern
    pattern = r'^[a-zA-Z0-9_]+$'

    # Use re.match to check if the input string matches the pattern
    match = re.match(pattern, input_str)

    # If there is a match, return True; otherwise, return False
    return bool(match)

# Test the function with some examples
test_strings = ["Valid_String_123", "invalid@string", "12345", ""]

for test_str in test_strings:
    if is_valid_string(test_str):
        print(f'The string "{test_str}" is valid.')
    else:
        print(f'The string "{test_str}" is invalid.')
```

```
The string "Valid_String_123" is valid.
The string "invalid@string" is invalid.
The string "12345" is valid.
The string "" is invalid.
```

In [33]:
```
# Question 12-Write a Python program where a string will start with a specific numb
#Answer:
def starts_with_number(input_str, target_number):
    # Convert the target number to a string for comparison
    target_str = str(target_number)

    # Use the startswith method to check if the input string starts with the target
    return input_str.startswith(target_str)

# Test the function with some examples
input_string = "12345HelloWorld"

target_number = 123

if starts_with_number(input_string, target_number):
    print(f'The string "{input_string}" starts with the number {target_number}.')
else:
    print(f'The string "{input_string}" does not start with the number {target_numb
```

```
The string "12345HelloWorld" starts with the number 123.
```

In [34]:
```
#Question 13- Write a Python program to remove leading zeros from an IP address
import re

def remove_leading_zeros(ip_address):
    # Define a regular expression pattern to match and remove leading zeros
    pattern = r'\b0+(\d+)\b'

    # Use re.sub to replace leading zeros with the captured digits
    result = re.sub(pattern, r'\1', ip_address)

    return result

# Test the function with an example
ip_address_with_zeros = "192.001.001.010"
ip_address_without_zeros = remove_leading_zeros(ip_address_with_zeros)
```

```
print(f'Original IP Address: {ip_address_with_zeros}')
print(f'IP Address without Leading Zeros: {ip_address_without_zeros}')
```

```
Original IP Address: 192.001.001.010
IP Address without Leading Zeros: 192.1.1.10
```

In [35]:
```
# Question 15- Write a Python program to search some literals strings in a string.
# Sample text : 'The quick brown fox jumps over the lazy dog.'
# Searched words : 'fox', 'dog', 'horse'

def search_literals(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 words in the text
found_words = search_literals(sample_text, searched_words)

# Display the result
for word in found_words:
    print(f'The word "{word}" was found in the text.')
```

```
The word "fox" was found in the text.
The word "dog" was found in the text.
```

In [36]:
```
#Question 16- Write a Python program to search a literals string in a string and al
#Sample text : 'The quick brown fox jumps over the lazy dog.'
#Searched words : 'fox'

def search_string(original_string, searched_word):
    # Using find() to search for the word in the string
    location = original_string.find(searched_word)

    if location != -1:
        print(f'The word "{searched_word}" was found in the string.')
        print(f'It starts at index {location} in the original string.')
    else:
        print(f'The word "{searched_word}" was not found in the string.')

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

# Searched word
searched_word = 'fox'

# Calling the function
search_string(sample_text, searched_word)
```

```
The word "fox" was found in the string.
It starts at index 16 in the original string.
```

In [37]:
```
# Question 17- Write a Python program to find the substrings within a string.
# Sample text : 'Python exercises, PHP exercises, C# exercises'
# Pattern : 'exercises'.

def find_substrings(original_string, pattern):
    start_index = 0
    occurrences = []
```

```
    while start_index < len(original_string):
        index = original_string.find(pattern, start_index)

        if index == -1:
            break

        occurrences.append(index)
        start_index = index + 1

    return occurrences

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

# Pattern to search for
pattern = 'exercises'

# Finding substrings
result = find_substrings(sample_text, pattern)

if result:
    print(f'The pattern "{pattern}" was found in the string at the following indice
    print(result)
else:
    print(f'The pattern "{pattern}" was not found in the string.')
```

```
The pattern "exercises" was found in the string at the following indices:
[7, 22, 36]
```

In [38]:

```
# Question 18- Write a Python program to find the occurrence and position of the su
def find_substrings_occurrences(original_string, pattern):
    start_index = 0
    occurrences = []

    while start_index < len(original_string):
        index = original_string.find(pattern, start_index)

        if index == -1:
            break

        occurrences.append(index)
        start_index = index + 1

    return occurrences

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

# Pattern to search for
pattern = 'exercises'

# Finding substrings and occurrences
occurrences = find_substrings_occurrences(sample_text, pattern)

if occurrences:
    print(f'The pattern "{pattern}" was found in the string at the following occurr
    for i, occurrence in enumerate(occurrences, 1):
        print(f'Occurrence {i}: Position {occurrence}')
else:
    print(f'The pattern "{pattern}" was not found in the string.')
```

```
The pattern "exercises" was found in the string at the following occurrences and p
ositions:
Occurrence 1: Position 7
Occurrence 2: Position 22
Occurrence 3: Position 36
```

In [39]:
```python
# Question 19- Write a Python program to convert a date of yyyy-mm-dd format to dd-
from datetime import datetime

def convert_date(input_date):
    # Convert string to datetime object
    input_date_object = datetime.strptime(input_date, "%Y-%m-%d")

    # Format the date in dd-mm-yyyy format
    output_date = input_date_object.strftime("%d-%m-%Y")

    return output_date

# Example usage
input_date_str = "2023-12-16"
output_date_str = convert_date(input_date_str)

print(f"Input date: {input_date_str}")
print(f"Output date: {output_date_str}")
```

```
Input date: 2023-12-16
Output date: 16-12-2023
```

In [40]:
```python
#Question 20- Create a function in python to find all decimal numbers with a precis
#The use of the re.compile() method is mandatory.
#Sample Text: "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"
#Expected Output: ['01.12', '145.8', '3.01', '27.25', '0.25']
import re

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

    # Find all matches in the input string
    matches = pattern.findall(input_string)

    return matches

# Sample text
sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

# Call the function and print the result
result = find_decimal_numbers(sample_text)
print(result)
```

```
['01.12', '145.8', '3.01', '27.25', '0.25']
```

In [41]:
```python
# Question 21- Write a Python program to separate and print the numbers and their p
def separate_numbers_with_positions(input_string):
    numbers = []

    for index, char in enumerate(input_string):
        if char.isdigit():
            # If the character is a digit, add it to the numbers list along with it
            numbers.append((char, index + 1))  # Adding 1 to index to make positior

    return numbers

# Example usage
input_string = "abc123def456ghi789"
```

```
result = separate_numbers_with_positions(input_string)

# Print the result
for number, position in result:
    print(f"Number: {number}, Position: {position}")
```

```
Number: 1, Position: 4
Number: 2, Position: 5
Number: 3, Position: 6
Number: 4, Position: 10
Number: 5, Position: 11
Number: 6, Position: 12
Number: 7, Position: 16
Number: 8, Position: 17
Number: 9, Position: 18
```

In [42]:

```
# Question 22- Write a regular expression in python program to extract maximum/larg
# Sample Text:  'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'
# Expected Output: 950

import re

def extract_maximum_numeric_value(input_string):
    # Define the regular expression pattern to match numeric values
    pattern = re.compile(r'\b\d+\b')

    # Find all matches in the input string
    matches = pattern.findall(input_string)

    # Convert the matched values to integers and find the maximum
    if matches:
        max_value = max(map(int, matches))
        return max_value
    else:
        return None

# Sample text
sample_text = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'

# Call the function and print the result
result = extract_maximum_numeric_value(sample_text)
print(result)
```

```
950
```

In [43]:

```
# Question 23- Create a function in python to insert spaces between words starting
# Sample Text: “RegularExpressionIsAnImportantTopicInPython"
# Expected Output: Regular Expression Is An Important Topic In Python

import re

def insert_spaces(input_string):
    # Use regular expression to insert spaces before capital letters
    spaced_string = re.sub(r'([a-z])([A-Z])', r'\1 \2', input_string)

    # Add a space before the first capital letter
    spaced_string = re.sub(r'([A-Z][a-z])', r' \1', spaced_string)

    return spaced_string

# Sample text
sample_text = "RegularExpressionIsAnImportantTopicInPython"

# Call the function and print the result
```

```
result = insert_spaces(sample_text)
print(result)
```

```
 Regular  Expression  Is  An  Important  Topic  In  Python
```

In [44]:
```
# Question 24- Python regex to find sequences of one upper case letter followed by
import re

text = "Aa Bb Cc Ab Cd Ef Gh"

pattern = re.compile(r'[A-Z][a-z]+')

matches = pattern.findall(text)

print(matches)
```

```
['Aa', 'Bb', 'Cc', 'Ab', 'Cd', 'Ef', 'Gh']
```

In [53]:
```
# Question 25- Write a Python program to remove continuous duplicate words from Ser
# Sample Text: "Hello hello world world"
# Expected Output: Hello hello world

import re

def remove_continuous_duplicates(sentence):
    # Use a regular expression to find continuous duplicate words
    pattern = re.compile(r'\b(\w+)(?:\s+\1\b)+', flags=re.IGNORECASE)

    # Replace continuous duplicate words with a single occurrence
    result = pattern.sub(r'\1', sentence)

    return result

# Sample Text
sample_text = "Hello hello world world"

# Remove continuous duplicate words
output = remove_continuous_duplicates(sample_text)

# Display the result
#print("Sample Text:", sample_text)
print("Output:", output)
```

```
Output: Hello world
```

In [46]:
```
# Question 26-  Write a python program using RegEx to accept string ending with alp
import re

def is_string_ending_with_alphanumeric(input_string):
    # Define a regular expression for a string ending with an alphanumeric characte
    pattern = re.compile(r'^.*[a-zA-Z0-9]$')

    # Test if the input string matches the pattern
    match = pattern.match(input_string)

    # Return True if there is a match, otherwise False
    return bool(match)

# Example usage
input_string = "Hello123"

if is_string_ending_with_alphanumeric(input_string):
    print(f'The string "{input_string}" ends with an alphanumeric character.')
else:
    print(f'The string "{input_string}" does not end with an alphanumeric character
```

```
The string "Hello123" ends with an alphanumeric character.
```

In [52]:
```
# Question 27-Write a python program using RegEx to extract the hashtags.
# Sample Text:  """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonet
# Expected Output: ['#Doltiwal', '#xyzabc', '#Demonetization']
import re

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

    # Find all matches in the text
    hashtags = pattern.findall(text)

    return hashtags

# Sample Text
sample_text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetiz

# Extract hashtags
output = extract_hashtags(sample_text)

# Display the result
#print("Sample Text:", sample_text)
print("Output:", output)
```

```
Output: ['#Doltiwal', '#xyzabc', '#Demonetization']
```

In [51]:
```
# Question 28- Write a python program using RegEx to remove <U+..> like symbols
# Check the below sample text, there are strange symbols something of the sort <U+.
# You need to come up with a general Regex expression that will cover all such symb
#Sample Text: "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+00
#              Those who  are protesting #demonetization  are all different party l
#Expected Output: @Jags123456 Bharat band on 28??<ed><ed>
#              Those who  are protesting #demonetization  are all different party l

import re

def remove_unicode_symbols(text):
    # Define a regular expression for removing <U+..> like symbols
    pattern = re.compile(r'<U\+[0-9A-Fa-f]+>')

    # Replace all matches with an empty string
    result = pattern.sub('', text)

    return result

# Sample Text
sample_text = "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+00

# Remove unicode symbols
output = remove_unicode_symbols(sample_text)

# Display the result
#print("Sample Text:", sample_text)
print("Output:", output)
```

```
Output: @Jags123456 Bharat band on 28??<ed><ed>Those who  are protesting #demoneti
zation  are all different party leaders
```

In [54]:
```
# Question 30- Create a function in python to remove all words from a string of len
#The use of the re.compile() method is mandatory.
#Sample Text: "The following example creates an ArrayList with a capacity of 50 ele
#4 elements are then added to the ArrayList and the ArrayList is trimmed accordingl
#Expected Output:  following example creates ArrayList a capacity elements.
```

```
# 4 elements added ArrayList ArrayList trimmed accordingly.

import re

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

    # Replace all matches with an empty string
    result = pattern.sub('', input_string)

    return result

# Sample Text
sample_text = "The following example creates an ArrayList with a capacity of 50 ele

# Remove words of length between 2 and 4
output = remove_words_of_length_between_2_and_4(sample_text)

# Display the result
# print("Sample Text:", sample_text)
print("Output:", output)
```

```
Output:  following example creates  ArrayList  a capacity   elements. 4 elements
added   ArrayList   ArrayList  trimmed accordingly.
```

In [ ]: