1- Write a Python program to replace all occurrences of a space, comma, or dot with a colon. Sample Text- 'Python Exercises, PHP exercises.' Expected Output: Python:Exercises::PHP:exercises:

In [3]:
```
sample_text = 'Python Exercises, PHP exercises.'
result = sample_text.replace(' ', ':').replace(',', ':').replace('.', ':')

print("Original text:", sample_text)
print("Modified text:", result)
```

```
Original text: Python Exercises, PHP exercises.
Modified text: Python:Exercises::PHP:exercises:
```

2.Create a dataframe using the dictionary below and remove everything (commas (,), !, XXXX, ;, etc.) from the columns except words. Dictionary- {'SUMMARY' : ['hello, world!', 'XXXXX test', '123four, five:; six...']} Expected output- 0 hello world 1 test 2 four five six

In [6]:
```
import pandas as pd
import re

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

df = pd.DataFrame(data)

def clean_text(text):
    cleaned_text = re.sub(r'[^a-zA-Z ]', '', text)
    return cleaned_text

df['SUMMARY'] = df['SUMMARY'].apply(clean_text)

print(df)
```

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

3.Create a function in python to find all words that are at least 4 characters long in a string. The use of the re.compile() method is mandatory.

In [7]:
```
import re

def find_long_words(input_string):
    pattern = re.compile(r'\b\w{4,}\b')

    result = pattern.findall(input_string)

    return result

input_string = "This is a sample string with some words like Python, exercise, and regex."
long_words = find_long_words(input_string)

print("Input string:", input_string)
print("Words with at least 4 characters:", long_words)
```

```
Input string: This is a sample string with some words like Python, exercise, and regex.
Words with at least 4 characters: ['This', 'sample', 'string', 'with', 'some', 'words', 'like', 'Python', 'ex
ercise', 'regex']
```

4- Create a function in python to find all three, four, and five character words in a string. The use of the re.compile() method is mandatory.

In [9]:
```
import re

def find_specific_length_words(input_string):
    pattern = re.compile(r'\b\w{3,5}\b')


    specific_length_words = pattern.findall(input_string)

    return specific_length_words


input_string = "This is a sample string with words of various lengths like hello, world, test, and more."
result = find_specific_length_words(input_string)
```

```
print("Input string:", input_string)
print("Three, four, and five-character words:", result)
```

```
Input string: This is a sample string with words of various lengths like hello, world, test, and more.
Three, four, and five-character words: ['This', 'with', 'words', 'like', 'hello', 'world', 'test', 'and', 'mo
re']
```

5- Create a function in Python to remove the parenthesis in a list of strings. The use of the re.compile() method is mandatory. Sample Text: ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data Science World)", "Data (Scientist)"]

In [10]:
```
import re

def remove_parentheses(strings_list):

    pattern = re.compile(r'\([^)]*\)')


    cleaned_list = [pattern.sub('', s) for s in strings_list]

    return cleaned_list

sample_text = [
    "example (.com)",
    "hr@fliprobo (.com)",
    "github (.com)",
    "Hello (Data Science World)",
    "Data (Scientist)"
]

result = remove_parentheses(sample_text)

print("Original list:", sample_text)
print("List after removing parentheses:", result)
```

```
Original list: ['example (.com)', 'hr@fliprobo (.com)', 'github (.com)', 'Hello (Data Science World)', 'Data
(Scientist)']
List after removing parentheses: ['example ', 'hr@fliprobo ', 'github ', 'Hello ', 'Data ']
```

6- Write a python program to remove the parenthesis area from the text stored in the text file using Regular Expression. Sample Text: ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data Science World)", "Data (Scientist)"] Expected Output: ["example", "hr@fliprobo", "github", "Hello", "Data"] Note- Store given sample text in the text file and then to remove the parenthesis area from the text.

In [12]:
```
import re

def remove_parentheses_from_file(input_filename, output_filename):
    with open(input_filename, 'r') as file:
        text = file.read()


    pattern = re.compile(r'\([^)]*\)')


    cleaned_text = pattern.sub('', text)


    with open(output_filename, 'w') as file:
        file.write(cleaned_text)

input_filename = 'input.txt'
output_filename = 'output.txt'

sample_text = [
    "example (.com)",
    "hr@fliprobo (.com)",
    "github (.com)",
    "Hello (Data Science World)",
    "Data (Scientist)"
]


with open(input_filename, 'w') as file:
    file.write("\n".join(sample_text))


remove_parentheses_from_file(input_filename, output_filename)
```

```
with open(output_filename, 'r') as file:
    result = file.read()

print("Original text in the file:")
print("\n".join(sample_text))
print("\nText after removing parentheses:")
print(result)
```

```
Original text in the file:
example (.com)
hr@fliprobo (.com)
github (.com)
Hello (Data Science World)
Data (Scientist)

Text after removing parentheses:
example
hr@fliprobo
github
Hello
Data
```

7- Write a regular expression in Python to split a string into uppercase letters. Sample text: “ImportanceOfRegularExpressionsInPython” Expected Output: [‘Importance’, ‘Of’, ‘Regular’, ‘Expression’, ‘In’, ‘Python’]

In [13]:
```
import re

sample_text = "ImportanceOfRegularExpressionsInPython"

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

result = pattern.split(sample_text)

print("Sample text:", sample_text)
print("Expected Output:", result)
```

```
Sample text: ImportanceOfRegularExpressionsInPython
Expected Output: ['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

8- Create a function in python to insert spaces between words starting with numbers. Sample Text: “RegularExpression1IsAn2ImportantTopic3InPython" Expected Output: RegularExpression 1IsAn 2ImportantTopic 3InPython

In [15]:
```
import re

def insert_spaces(text):
    pattern = re.compile(r'(?<=\d)(?=\D)')

    modified_text = pattern.sub(' ', text)

    return modified_text

sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
result = insert_spaces(sample_text)

print("Sample text:", sample_text)
print("Expected Output:", result)
```

```
Sample text: RegularExpression1IsAn2ImportantTopic3InPython
Expected Output: RegularExpression1 IsAn2 ImportantTopic3 InPython
```

Question 9- Create a function in python to insert spaces between words starting with capital letters or with numbers. Sample Text: “RegularExpression1IsAn2ImportantTopic3InPython" Expected Output: RegularExpression 1 IsAn 2 ImportantTopic 3 InPython

In [17]:
```
import re

def insert_spaces(text):
    pattern = re.compile(r'(?<=[A-Z0-9])(?=[^A-Z0-9])')

    modified_text = pattern.sub(' ', text)

    return modified_text

sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
result = insert_spaces(sample_text)
```

```
print("Sample text:", sample_text)
print("Expected Output:", result)
```

```
Sample text: RegularExpression1IsAn2ImportantTopic3InPython
Expected Output: R egularE xpression1I sA n2I mportantT opic3I nP ython
```

10- Use the github link below to read the data and create a dataframe. After creating the dataframe extract the first 6 letters of each country and store in the dataframe under a new column called first_five_letters.

In [18]:
```
import pandas as pd

github_link = "https://raw.githubusercontent.com/dsrscientist/DSData/master/happiness_score_dataset.csv"

df = pd.read_csv(github_link)

df['first_six_letters'] = df['Country'].str[:6]

print(df.head())
```

```
       Country          Region  Happiness Rank  Happiness Score  \
0  Switzerland  Western Europe               1            7.587
1      Iceland  Western Europe               2            7.561
2      Denmark  Western Europe               3            7.527
3       Norway  Western Europe               4            7.522
4       Canada   North America               5            7.427

   Standard Error  Economy (GDP per Capita)   Family  \
0         0.03411                   1.39651  1.34951
1         0.04884                   1.30232  1.40223
2         0.03328                   1.32548  1.36058
3         0.03880                   1.45900  1.33095
4         0.03553                   1.32629  1.32261

   Health (Life Expectancy)  Freedom  Trust (Government Corruption)  \
0                   0.94143  0.66557                        0.41978
1                   0.94784  0.62877                        0.14145
2                   0.87464  0.64938                        0.48357
3                   0.88521  0.66973                        0.36503
4                   0.90563  0.63297                        0.32957

   Generosity  Dystopia Residual first_six_letters
0     0.29678            2.51738            Switze
1     0.43630            2.70201            Icelan
2     0.34139            2.49204            Denmar
3     0.34699            2.46531            Norway
4     0.45811            2.45176            Canada
```

11- Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores.

In [19]:
```
import re

def is_valid_string(input_string):
    pattern = re.compile("^[a-zA-Z0-9_]+$")
    return bool(pattern.match(input_string))
input_string = input("Enter a string: ")
if is_valid_string(input_string):
    print("The string is valid.")
else:
    print("The string is not valid.")
```

```
Enter a string: 10
The string is valid.
```

12- Write a Python program where a string will start with a specific number.

In [21]:
```
def starts_with_number(input_string, start_number):
    return input_string.startswith(str(start_number))

input_string = input("Enter a string: ")
start_number = input("Enter the specific number: ")

if starts_with_number(input_string, start_number):
    print(f"The string starts with {start_number}.")
else:
    print(f"The string does not start with {start_number}.")
```

```
Enter a string: "fliprobo"
Enter the specific number: intership
The string does not start with intership.
```

13-Write a Python program to remove leading zeros from an IP address

In [24]:
```python
def remove_leading_zeros(ip_address):
    # Split the IP address into segments
    segments = ip_address.split('.')

    cleaned_segments = [str(int(segment)) for segment in segments]

    cleaned_ip_address = '.'.join(cleaned_segments)

    return cleaned_ip_address

ip_address = input("Enter an IP address: ")
cleaned_ip = remove_leading_zeros(ip_address)
print(f"The cleaned IP address is: {cleaned_ip}")
```

```
Enter an IP address: 8.4.0.8.3
The cleaned IP address is: 8.4.0.8.3
```

Question 14- Write a regular expression in python to match a date string in the form of Month name followed by day number and year stored in a text file. Sample text : ' On August 15th 1947 that India was declared independent from British colonialism, and the reins of control were handed over to the leaders of the Country'.

In [29]:
```python
import re

sample_text = 'On August 15th 1947 that India was declared independent from British colonialism, and the rein

date_pattern = re.compile(r'\b(?:January|February|March|April|May|June|July|August|September|October|November

matches = date_pattern.findall(sample_text)

if matches:
    print("Dates found in the sample text:")
    for date in matches:
        print(date)
else:
    print("No dates found in the sample text.")
```

```
Dates found in the sample text:
August 15th 1947
```

In [28]:
```python
14- Write a regular expression in python to match a date string in the form of Month name followed by day num
```

```
  Cell In[28], line 1
    14- Write a regular expression in python to match a date string in the form of Month name followed by day
number and year stored in a text file.
                ^
SyntaxError: invalid syntax
```

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'

In [30]:
```python
def search_literals(main_string, search_words):
    found_words = []
    for word in search_words:
        if word in main_string:
            found_words.append(word)
    return found_words

sample_text = 'The quick brown fox jumps over the lazy dog.'
searched_words = ['fox', 'dog', 'horse']

result = search_literals(sample_text, searched_words)

print(f"Sample text: '{sample_text}'")
print(f"Searched words: {searched_words}")
print(f"Found words: {result}")
```

```
Sample text: 'The quick brown fox jumps over the lazy dog.'
Searched words: ['fox', 'dog', 'horse']
Found words: ['fox', 'dog']
```

16- Write a Python program to search a literals string in a string and also find the location within the original string where the pattern occurs Sample text : 'The quick brown fox jumps over the lazy dog.' Searched words : 'fox'

In [32]:
```
def search_literal_with_location(main_string, search_word):
    locations = []
    start_index = main_string.find(search_word)

    while start_index != -1:
        locations.append(start_index)
        start_index = main_string.find(search_word, start_index + 1)

    return locations

sample_text = 'The quick brown fox jumps over the lazy dog.'
searched_word = 'fox'

result = search_literal_with_location(sample_text, searched_word)

print(f"Sample text: '{sample_text}'")
print(f"Searched word: '{searched_word}'")
if result:
    print(f"Locations: {result}")
else:
    print(f"The word '{searched_word}' was not found in the sample text.")
```

```
Sample text: 'The quick brown fox jumps over the lazy dog.'
Searched word: 'fox'
Locations: [16]
```

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

In [34]:
```
def find_substrings(main_string, pattern):
    start_index = main_string.find(pattern)
    substrings = []

    while start_index != -1:
        substrings.append(main_string[start_index:start_index + len(pattern)])
        start_index = main_string.find(pattern, start_index + 1)

    return substrings

sample_text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'

result = find_substrings(sample_text, pattern)

print(f"Sample text: '{sample_text}'")
print(f"Pattern: '{pattern}'")
if result:
    print(f"Substrings: {result}")
else:
    print(f"No occurrences of the pattern '{pattern}' found in the sample text.")
```

```
Sample text: 'Python exercises, PHP exercises, C# exercises'
Pattern: 'exercises'
Substrings: ['exercises', 'exercises', 'exercises']
```

18-Write a Python program to find the occurrence and position of the substrings within a string.

In [35]:
```
def find_occurrences_and_positions(main_string, pattern):
    occurrences = []
    start_index = main_string.find(pattern)

    while start_index != -1:
        occurrences.append((pattern, start_index))
        start_index = main_string.find(pattern, start_index + 1)

    return occurrences

sample_text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'

result = find_occurrences_and_positions(sample_text, pattern)

print(f"Sample text: '{sample_text}'")
```

```
print(f"Pattern: '{pattern}'")

if result:
    print("Occurrences and Positions:")
    for occurrence, position in result:
        print(f"Occurrence: '{occurrence}', Position: {position}")
else:
    print(f"No occurrences of the pattern '{pattern}' found in the sample text.")
```

```
Sample text: 'Python exercises, PHP exercises, C# exercises'
Pattern: 'exercises'
Occurrences and Positions:
Occurrence: 'exercises', Position: 7
Occurrence: 'exercises', Position: 22
Occurrence: 'exercises', Position: 36
```

19- Write a Python program to convert a date of yyyy-mm-dd format to dd-mm-yyyy format.

In [36]:
```
from datetime import datetime

def convert_date_format(input_date):
    input_date_object = datetime.strptime(input_date, '%Y-%m-%d')

    output_date = input_date_object.strftime('%d-%m-%Y')

    return output_date

input_date = '2022-01-31'
output_date = convert_date_format(input_date)

print(f"Original date: {input_date}")
print(f"Converted date: {output_date}")
```

```
Original date: 2022-01-31
Converted date: 31-01-2022
```

20- Create a function in python to find all decimal numbers with a precision of 1 or 2 in a string. 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']

In [37]:
```
import re

def find_decimal_numbers(text):
    pattern = re.compile(r'\b\d+\.\d{1,2}\b')

    matches = pattern.findall(text)

    return matches

sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

result = find_decimal_numbers(sample_text)

print(f"Sample Text: '{sample_text}'")
print(f"Expected Output: {result}")
```

```
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']
```

21- Write a Python program to separate and print the numbers and their position of a given string.

In [39]:
```
import re

def separate_numbers_and_positions(input_string):
    pattern = re.compile(r'\d+')

    matches = pattern.finditer(input_string)

    for match in matches:
        number = match.group()
        position = match.span()
        print(f"Number: {number}, Position: {position}")

sample_text = 'The price is $45.50 and the quantity is 10.'
separate_numbers_and_positions(sample_text)
```

```
Number: 45, Position: (14, 16)
Number: 50, Position: (17, 19)
Number: 10, Position: (40, 42)
```

22-Write a regular expression in python program to extract maximum/largest numeric value from a string. Sample Text: 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642' Expected Output: 950

In [41]:
```python
import re

def extract_maximum_numeric_value(input_string):
    pattern = re.compile(r'\b\d+\b')


    numeric_values = [int(match) for match in pattern.findall(input_string)]

    max_numeric_value = max(numeric_values)

    return max_numeric_value

# 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
```

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

In [65]:
```python
import re

def insert_spaces_between_capital_words(input_text):
    pattern = re.compile(r'(?<=[a-z])([A-Z])')

    spaced_text = pattern.sub(r' \1', input_text)

    spaced_text = spaced_text.capitalize()

    return spaced_text

sample_text = "RegularExpressionIsAnImportantTopicInPython"
result = insert_spaces_between_capital_words(sample_text)

print(f"Sample Text: '{sample_text}'")
print(f"Expected Output: {result}")
```

```
Sample Text: 'RegularExpressionIsAnImportantTopicInPython'
Expected Output: Regular expression is an important topic in python
```

24- Python regex to find sequences of one upper case letter followed by lower case letters

In [43]:
```python
import re

def find_sequences(input_text):
    pattern = re.compile(r'\b[A-Z][a-z]*\b')
    sequences = pattern.findall(input_text)
    return sequences

sample_text = "Find Sequences of One uppercase Letter Followed by lowercase letters in Python"
result = find_sequences(sample_text)

print(f"Sample Text: '{sample_text}'")
print(f"Found Sequences: {result}")
```

```
Sample Text: 'Find Sequences of One uppercase Letter Followed by lowercase letters in Python'
Found Sequences: ['Find', 'Sequences', 'One', 'Letter', 'Followed', 'Python']
```

25- Write a Python program to remove continuous duplicate words from Sentence using Regular Expression. Sample Text: "Hello hello world world" Expected Output: Hello hello world

In [46]:
```
import re

def remove_continuous_duplicates(sentence):
    pattern = re.compile(r'\b(\w+)(?:\s+\1\b)+', flags=re.IGNORECASE)

    cleaned_sentence = pattern.sub(r'\1', sentence)

    return cleaned_sentence

sample_text = "Hello hello world world"
result = remove_continuous_duplicates(sample_text)

print(f"Sample Text: '{sample_text}'")
print(f"Expected Output: {result}")
```

```
Sample Text: 'Hello hello world world'
Expected Output: Hello world
```

26- Write a python program using RegEx to accept string ending with alphanumeric character.

In [47]:
```
import re

def ends_with_alphanumeric(input_string):
    pattern = re.compile(r'\w$')  # \w matches any alphanumeric character, and $ asserts the end of the strin
    return bool(pattern.search(input_string))

test_string1 = "Hello123"
test_string2 = "World!"
test_string3 = "123456"

print(f"{test_string1} ends with alphanumeric: {ends_with_alphanumeric(test_string1)}")
print(f"{test_string2} ends with alphanumeric: {ends_with_alphanumeric(test_string2)}")
print(f"{test_string3} ends with alphanumeric: {ends_with_alphanumeric(test_string3)}")
```

```
Hello123 ends with alphanumeric: True
World! ends with alphanumeric: False
123456 ends with alphanumeric: True
```

27-Write a python program using RegEx to extract the hashtags. Sample Text: """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has rendered USELESS <U+00A0><U+00BD><U+00B1><U+0089> "acquired funds" No wo""" Expected Output: ['#Doltiwal', '#xyzabc', '#Demonetization']

In [49]:
```
import re

def extract_hashtags(input_text):
    pattern = re.compile(r'#\w+')
    hashtags = pattern.findall(input_text)
    return hashtags

# Example usage
sample_text = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has rend

result = extract_hashtags(sample_text)

print(f"Sample Text: '{sample_text}'")
print(f"Extracted Hashtags: {result}")
```

```
Sample Text: 'RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has rendere
d USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo'
Extracted Hashtags: ['#Doltiwal', '#xyzabc', '#Demonetization']
```

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+..> all over the place. You need to come up with a general Regex expression that will cover all such symbols. Sample Text: "@Jags123456 Bharat band on 28??<U+00A0><U+00BD><U+00B8><U+0082>Those who are protesting #demonetization are all different party leaders" Expected Output: @Jags123456 Bharat band on 28??Those who are protesting #demonetization are all different party leaders

In [51]:
```
import re

def remove_unicode_symbols(input_text):
    pattern = re.compile(r'<U\+\w+>')
    cleaned_text = pattern.sub('', input_text)
    return cleaned_text
```

```
sample_text = "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who  are protesti

result = remove_unicode_symbols(sample_text)

print(f"Sample Text: '{sample_text}'")
print(f"Expected Output: {result}")
```

```
Sample Text: '@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who  are protestin
g #demonetization  are all different party leaders'
Expected Output: @Jags123456 Bharat band on 28??<ed><ed>Those who  are protesting #demonetization  are all di
fferent party leaders
```

30- Create a function in python to remove all words from a string of length between 2 and 4. The use of the re.compile() method is mandatory. 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." Expected Output: following example creates ArrayList a capacity elements. 4 elements added ArrayList ArrayList trimmed accordingly.

In [64]:
```
import re

def remove_words_between_lengths(input_text):
    pattern = re.compile(r'\b\w{2,4}\b')
    cleaned_text = pattern.sub('', input_text)
    return cleaned_text

sample_text = "The following example creates an ArrayList with a capacity of 50 elements. 4 elements are then

result = remove_words_between_lengths(sample_text)

print(f"Sample Text: '{sample_text}'")
print(f"Expected Output: {result}")
```

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

In [ ]:
```

```

In [ ]:
```

```