Regular Expressions Assignment 3:

Question 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:

Answer:

import re

text ='Python Exercises, PHP exercises.'

pattern=r'[,.]'

result=re.sub(pattern,':',text)

print(result)

```
In [3]: import re
        text ='Python Exercises, PHP exercises.'
        pattern=r'[ ,.]'
        result=re.sub(pattern,':',text)
        print(result)

        Python:Exercises::PHP:exercises:

In [ ]:
```

Question 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

Answer:

```
In [10]: import pandas as pd
         import re
         input={'SUMMARY': ['hello, world!', 'XXXXX test', '123four, five:; six...']}
         df= pd.DataFrame(input)
         def clean_text(text):
             # Remove everything except alphabets and spaces
             cleaned_text = re.sub(r'[^a-zA-Z\s]', '', text)
             # Remove multiple spaces and strip leading/trailing spaces
             cleaned_text = re.sub(r'\s+', ' ', cleaned_text).strip()
             return cleaned_text

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

         # Print the cleaned DataFrame
         print(df)

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

Question 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.

Answer:

```
In [12]: import re

         def four_words_fn(input_string):
             pattern = re.compile(r'\b\w{4,}\b')
             four_words = pattern.findall(input)
             return four_words

         input= "This is a program written by sunanda for question four."
         result = four_words_fn(input)
         print(result)

         ['This', 'program', 'written', 'sunanda', 'question', 'four']

In [ ]:
```

Question 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 [13]: import re

         def four_words_fn(input_string):
             pattern = re.compile(r'\b\w{3,5}\b')
             four_words = pattern.findall(input)
             return four_words

         input= "This is a program written by sunanda for question five three four five letter words."
         result = four_words_fn(input)
         print(result)

         ['This', 'for', 'five', 'three', 'four', 'five', 'words']

In [ ]:
```

Question 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)"]

Expected Output:

example.com

hr@fliprobo.com

github.com

Hello Data Science World

Data Scientist

Answer:

```
import re

def remove_parentheses(string):
    pattern = re.compile(r'\s*\([^)]*\)|\s*\)')
    result = [pattern.sub('', s) for s in string]
    return result
```

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

final_result = remove_parentheses(input)

for text in final_result:

    print(text)

Question 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.

Answer:

In [23]:
```
import re


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


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


def remove_parenthesis_area(filename):
    cleaned_lines = []
    with open(filename, 'r') as file:
        for line in file:
            cleaned_line = re.sub(r'\s*\([^)]*\)', '', line)
            cleaned_lines.append(cleaned_line.strip())
    return cleaned_lines
```

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


def remove_parenthesis_area(filename):
    cleaned_lines = []
    with open(filename, 'r') as file:
        for line in file:
            cleaned_line = re.sub(r'\s*\([^)]*\)', '', line)
            cleaned_lines.append(cleaned_line.strip())
    return cleaned_lines

filename = 'sample_text.txt'
cleaned_text = remove_parenthesis_area(filename)


print(cleaned_text)
```

```
['example', 'hr@fliprobo', 'github', 'Hello', 'Data']
```

Question 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’]

Answer:

```
import re



input = "ImportanceOfRegularExpressionsInPython"



result = re.findall(r'[A-Z][a-z]*', input)



print(result)
```

Question 8- Create a function in python to insert spaces between words starting with numbers.

Sample Text: “RegularExpression1IsAn2ImportantTopic3InPython"

Expected Output: RegularExpression 1IsAn 2ImportantTopic 3InPython

Answer:

```
import re


def insert_spaces(text):
```

```
    modified_text = re.sub(r'([A-Z\d])([A-Z][a-z])', r'\1 \2', text)
    return modified_text


sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"


output_text = insert_spaces(sample_text)


print(output_text)
```

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

```
import re

def insert_spaces(text):
    modified_text = re.sub(r'([A-Z\d])([A-Z][a-z])', r'\1 \2', text)
    modified_text = re.sub(r'(\d)([A-Z])', r'\1 \2', modified_text)
    return modified_text


sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"


output_text = insert_spaces(sample_text)
print(output_text)
```

Question 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.

Github Link-
https://raw.githubusercontent.com/dsrscientist/DSData/master/happiness_score_dataset.csv

Answer:

import pandas as pd

url = 'https://raw.githubusercontent.com/dsrscientist/DSData/master/happiness_score_dataset.csv'

df = pd.read_csv(url)

print("Original DataFrame:")

print(df.head())

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

print("\nDataFrame with 'first_five_letters' column:")

print(df.head())

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

Answer:

```
In [30]: import re

def match_string(s):
    # Regular expression pattern
    pattern = r'^[a-zA-Z0-9_]+$'

    # Match the string against the pattern
    if re.match(pattern, s):
        return True
    else:
        return False

# Test cases
strings = [
    "Hello_World123",
    "PythonProgramming",
    "12345",
    "abc_def_456",
    "123_abc_456_xyz",
    "Special@Characters!"
]

# Check each string
for s in strings:
```

```
# Test cases
strings = [
    "Hello_World123",
    "PythonProgramming",
    "12345",
    "abc_def_456",
    "123_abc_456_xyz",
    "Special@Characters!"
]

# Check each string
for s in strings:
    if match_string(s):
        print(f"{s} matches the pattern.")
    else:
        print(f"{s} does not match the pattern.")
```

```
Hello_World123 matches the pattern.
PythonProgramming matches the pattern.
12345 matches the pattern.
abc_def_456 matches the pattern.
123_abc_456_xyz matches the pattern.
Special@Characters! does not match the pattern.
```

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

Answer:

In [31]:
```
import re

def starts_with_number(string, number):
    # Regular expression pattern to match string starting with a specific number
    pattern = re.compile(r'^' + re.escape(number) + r'.*')

    # Match the string against the pattern
    if re.match(pattern, string):
        return True
    else:
        return False

# Test cases
strings = [
    "123abc",
    "456def",
    "789xyz",
    "01234",
    "56789",
    "abc123",
    "def456",
    "xyz789"
]
```

```
]

number_to_match = "123"

# Check each string
for s in strings:
    if starts_with_number(s, number_to_match):
        print(f"{s} starts with {number_to_match}.")
    else:
        print(f"{s} does not start with {number_to_match}.")
```

```
123abc starts with 123.
456def does not start with 123.
789xyz does not start with 123.
01234 does not start with 123.
56789 does not start with 123.
abc123 does not start with 123.
def456 does not start with 123.
xyz789 does not start with 123.
```

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

Answer:

In [32]:
```
def remove_leading_zeros(ip_address):
    # Split the IP address into its parts (octets)
    octets = ip_address.split('.')

    # Convert each octet to integer and then back to string to remove leading zeros
    cleaned_octets = [str(int(octet)) for octet in octets]

    # Join the octets back into a single IP address string
    cleaned_ip = '.'.join(cleaned_octets)

    return cleaned_ip

# Test cases
ip_addresses = [
    "192.168.001.001",
    "010.001.002.003",
    "000.100.200.300",
    "255.000.000.000",
    "000.000.000.001"
]

# Remove leading zeros from each IP address
for ip in ip_addresses:
    cleaned_ip = remove_leading_zeros(ip)
```

```
    return cleaned_ip

# Test cases
ip_addresses = [
    "192.168.001.001",
    "010.001.002.003",
    "000.100.200.300",
    "255.000.000.000",
    "000.000.000.001"
]

# Remove leading zeros from each IP address
for ip in ip_addresses:
    cleaned_ip = remove_leading_zeros(ip)
    print(f"Original IP: {ip} | Cleaned IP: {cleaned_ip}")
```

```
Original IP: 192.168.001.001 | Cleaned IP: 192.168.1.1
Original IP: 010.001.002.003 | Cleaned IP: 10.1.2.3
Original IP: 000.100.200.300 | Cleaned IP: 0.100.200.300
Original IP: 255.000.000.000 | Cleaned IP: 255.0.0.0
Original IP: 000.000.000.001 | Cleaned IP: 0.0.0.1
```

In [ ]:

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'.

Expected Output- August 15th 1947

Note- Store given sample text in the text file and then extract the date string asked format.

import re

Read the text from the text file

with open('sample_text.txt', 'r') as file:

    text = file.read()

Define the regular expression pattern

pattern = r'\b([A-Z][a-z]+ \d{1,2}(?:st|nd|rd|th)? \d{4})\b'

Search for the pattern in the text

```
match = re.search(pattern, text)


if match:
    date_string = match.group(1)
    print("Extracted Date String:", date_string)
else:
    print("No date string matching the format was found.")
```

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'

In [38]:
```
def search_literals(main_string, searched_words):
    found_words = []
    for word in searched_words:
        if word in main_string:
            found_words.append(word)
    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 each word in the sample text
found_words = search_literals(sample_text, searched_words)

# Print the results
if found_words:
    print(f"Found words: {', '.join(found_words)}")
else:
    print("No matching words found.")
```

```
Found words: fox, dog
```

Question 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 [39]:
```
import re

def find_locations(main_string, searched_word):
    pattern = re.compile(re.escape(searched_word))
    matches = pattern.finditer(main_string)
    locations = [match.start() for match in matches]
    return locations

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

# Searched word
searched_word = 'fox'

# Find all occurrences and their locations using regular expressions
locations = find_locations(sample_text, searched_word)

# Print the results
if locations:
    print(f"'{searched_word}' found at location(s): {locations}")
else:
    print(f"'{searched_word}' not found in the text.")
```

```
'fox' found at location(s): [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 [40]:
```
import re

def find_substrings(main_string, pattern):
    matches = re.finditer(pattern, main_string)
    locations = [match.start() for match in matches]
    return locations

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

# Pattern to search for
pattern = r'exercises'

# Find all occurrences and their locations using regular expressions
locations = find_substrings(sample_text, pattern)

# Print the results
if locations:
    print(f"Pattern '{pattern}' found at location(s): {locations}")
else:
    print(f"Pattern '{pattern}' not found in the text.")
```

```
Pattern 'exercises' found at location(s): [7, 22, 36]
```

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

Answer:

In [41]:
```
import re

def find_occurrences_positions_regex(main_string, substring):
    pattern = re.compile(re.escape(substring))
    matches = pattern.finditer(main_string)
    positions = [match.start() for match in matches]
    return positions

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

# Substring to search for
substring = 'exercises'

# Find occurrences and their positions using regular expressions
positions = find_occurrences_positions_regex(sample_text, substring)
occurrence_count = len(positions)

# Print the results
print(f"Substring '{substring}' occurs {occurrence_count} time(s) at position(s): {positions}")
```

```
Substring 'exercises' occurs 3 time(s) at position(s): [7, 22, 36]
```

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

Answer:

In [42]:
```
import re

def convert_date_format(date_string):

    pattern = re.compile(r'(\d{4})-(\d{2})-(\d{2})')


    result = re.sub(pattern, r'\3-\2-\1', date_string)

    return result


date_in_yyyy_mm_dd = '2023-07-13'
converted_date = convert_date_format(date_in_yyyy_mm_dd)

print(f'Original date: {date_in_yyyy_mm_dd}')
print(f'Converted date: {converted_date}')
```

```
Original date: 2023-07-13
Converted date: 13-07-2023
```

In [ ]:

Question 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']

Answer:

In [43]:
```
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
sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

# Find decimal numbers with precision of 1 or 2
decimal_numbers = find_decimal_numbers(sample_text)

# Print the results
print("Original Text:", sample_text)
print("Decimal Numbers with precision of 1 or 2:", decimal_numbers)
```

```
Original Text: 01.12 0132.123 2.31875 145.8 3.01 27.25 0.25
Decimal Numbers with precision of 1 or 2: ['01.12', '145.8', '3.01', '27.25', '0.25']
```

In [ ]: