Regular Expressions

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:

Ans : text = text.replace(' ', ':').replace(',', ':').replace('.', ':') # Print the modified text print ('Python Exercises, PHP exercises.')

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

Ans : import pandas as pd

import re

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

df = pd.DataFrame(data)

text(text):

text = re.sub(r'[^a-zA-Z\s]', '', text)

text = re.sub(r'\s+', ' ', text).strip() return text df['SUMMARY'] = df['SUMMARY'].apply(clean_text)

print(df)

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.

Ans : import re

```
def find_long_words(text):

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

matches = pattern.findall(text)

 return matches

 text = "Here are some examples of words: apple, bananas, and oranges.

long_words = find_long_words(text)

print(long_words)
```

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.

Ans : import re

```
 def find_specific_length_words(text):

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

matches = pattern.findall(text)

matches = [word for word in matches if 3 <= len(word) <= 5] return filtered_matches

text = "Here are some examples of small words: cat, dogs, apple, and bananas." specific_length_words =
find_specific_length_words(text) print(specific_length_words)
```

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
```

Ans : import re

def remove_parentheses(strings_list):

pattern = re.compile(r'\s*\(.*?\)')

cleaned_strings = [pattern.sub('', s).strip() for s in strings_list] return cleaned_strings

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

output = remove_parentheses(sample_text) print(output)

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.

Ans : import re

def remove_parentheses_from_file(input_file, output_file):

pattern = re.compile(r'\s*\(.*?\)')

with open(input_file, 'r') as file: lines = file.readlines()

cleaned_lines = [pattern.sub('', line).strip() for line in lines]

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

```
input_file = 'sample.txt' output_file = 'cleaned_sample.txt'

remove_parentheses_from_file(input_file, output_file)

print(f"Processed text has been saved to {output_file}")
```

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’]

Ans : import re

```
def split_by_uppercase(text):

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

 matches = pattern.findall(text)

return matches

text = "ImportanceOfRegularExpressionsInPython"

result = split_by_uppercase(text)

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

Ans : import re def insert_spaces(text):

```
pattern = re.compile(r'(\d)([A-Za-z])')

result = pattern.sub(r'\1 \2', text) return result

text = "RegularExpression1IsAn2ImportantTopic3InPython"

output = insert_spaces(text)

print(output)
```

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

Ans : import re def insert_spaces(text):

```
pattern = re.compile(r'(?<=[A-Z0-9])(?=[a-z0-9])')

result = pattern.sub(' ', text) return result

text = "RegularExpression1IsAn2ImportantTopic3InPython"

output = insert_spaces(text)

 print(output)
```

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

Ans : 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_six_letters'] = df['Country'].apply(lambda x: x[:6])

 print("\nDataFrame with first_six_letters:")

print(df.head())
```

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

Ans : import re def match_string(s):

```
pattern = r'^[a-zA-Z0-9_]+$'

 # Use re.match() to check if the string matches the pattern if re.match(pattern, s): return True else: return
False

# Test cases test_strings = [ "ValidString123_",

# Should match "Invalid String!",
# Should not match (contains space and exclamation mark) "AnotherValid_456",

# Should match "Invalid@String",

 # Should not match (contains @ symbol) "Valid_string_with_numbers_123"

 # Should match ]

for s in test_strings: result = match_string(s)

 print(f"'{s}' matches: {result}")
```

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

Ans : import re

```
def starts_with_number(s, number):

    # Define the regular expression pattern to check if the string starts with the specified number

    pattern = r'^' + re.escape(str(number))

        # Use re.match() to check if the string starts with the specified number

    if re.match(pattern, s):

        return True

    else:

        return False

# Test cases

test_strings = [

    "123abc",    # Should match if number is 123
    "456xyz",    # Should match if number is 456
    "789",       # Should match if number is 789
    "abc123",    # Should not match if number is 123
    "1234hello"  # Should match if number is 1234
]
number_to_check = 123
# Check each test string
for s in test_strings:
    result = starts_with_number(s, number_to_check)
    print(f"'{s}' starts with {number_to_check}: {result}")
```

Using String Manipulation

```
def starts_with_number(s, number):
    # Convert number to string for comparison
    number_str = str(number)
        # Check if the string starts with the specified number
    if s.startswith(number_str):
        return True
    else:
        return False
```

```
# Test cases

test_strings = [

    "123abc",    # Should match if number is 123

    "456xyz",    # Should match if number is 456

    "789",       # Should match if number is 789

    "abc123",    # Should not match if number is 123

    "1234hello"  # Should match if number is 1234

]

number_to_check = 123

# Check each test string

for s in test_strings:

    result = starts_with_number(s, number_to_check)

    print(f"'{s}' starts with {number_to_check}: {result}")
```

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

Ans : def remove_leading_zeros(ip_address):

```
    # Split the IP address into its octets

    octets = ip_address.split('.')

        # Remove leading zeros from each octet

    cleaned_octets = [str(int(octet)) for octet in octets]
    # Join the cleaned octets back into an IP address

    cleaned_ip_address = '.'.join(cleaned_octets)

        return cleaned_ip_address

# Test cases

test_ips = [

    "192.168.001.001",  # Should become "192.168.1.1"

    "10.000.0.255",     # Should become "10.0.0.255"

    "255.255.255.255",  # Should remain "255.255.255.255"

    "001.002.003.004",  # Should become "1.2.3.4"

    "000.0.0.000"       # Should become "0.0.0.0"

]



# Process and print each test IP address

for ip in test_ips:

    cleaned_ip = remove_leading_zeros(ip)

    print(f"Original IP: {ip} => Cleaned IP: {cleaned_ip}")
```

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.

Ans : import re

```
def extract_date_from_file(filename):

    # Define the regular expression pattern for the date

    # Pattern explanation:

    # - Month name (full name, e.g., August)

    # - Followed by a space and a day number (1-31)

    # - Optionally followed by 'st', 'nd', 'rd', or 'th'

    # - Followed by a year (4 digits)

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

    with open(filename, 'r') as file:

        text = file.read()

            # Search for the date pattern in the text

    match = re.search(pattern, text)

        if match:

        # Format the date string in the desired format

        date_str = f"{match.group(1)} {match.group(2)} {match.group(3)}"

        return date_str
```

```
  else:

        return "No date found"

# File path

filename = 'sample_text.txt'

# Extract and print the date

extracted_date = extract_date_from_file(filename)

print("Extracted Date:", extracted_date)
```

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'

Ans : def search_literals_in_text(text, search_words):

```
    # Initialize a dictionary to store the search results

    results = {}

    # Loop through each search word and check if it's in the text

    for word in search_words:

        if word in text:

            results[word] = True

        else:

            results[word] = False
    return results

# Sample text

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

# Words to search for

search_words = ['fox', 'dog', 'horse']

# Perform the search

search_results = search_literals_in_text(sample_text, search_words)

# Print the results

for word, found in search_results.items():

    if found:

        print(f"'{word}' is present in the text.")

    else:

        print(f"'{word}' is not present in the text.")
```

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'

Ans : def search_literal_with_locations(text, search_word):

```
    # Initialize the starting position and an empty list for results

    start = 0

    locations = []

    # Search for the search_word in the text

    while True:

        start = text.find(search_word, start)

        if start == -1:

            break

        locations.append(start)

        start += len(search_word)  # Move past the current match

    return locations

# Sample text

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

# Word to search for

search_word = 'fox'

# Perform the search

locations = search_literal_with_locations(sample_text, search_word)

# Print results

if locations:

    for loc in locations:
        print(f"'{search_word}' found at index {loc}.")
else:
    print(f"'{search_word}' not found in the text.")
```

Question 17- Write a Python program to find the substrings within a string.

Sample text : 'Python exercises, PHP exercises, C# exercises'

Pattern : 'exercises'.

Ans : import re

```
def find_substrings(text, pattern):
    locations = []
    for match in re.finditer(re.escape(pattern), text):
        locations.append(match.start())
    return locations
# Sample text
sample_text = 'Python exercises, PHP exercises, C# exercises'
# Pattern to search for
pattern = 'exercises'
# Find all occurrences
locations = find_substrings(sample_text, pattern)
```

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

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

Ans : def find_substring_occurrences(text, pattern):

```
    start = 0
    occurrences = []
    while True:
        start = text.find(pattern, start)
        if start == -1:
            break
        occurrences.append(start)
        start += len(pattern)  # Move past the current match
    return occurrences
# Sample text
sample_text = 'Python exercises, PHP exercises, C# exercises'
```

```
# Substring to search for

pattern = 'exercises'

# Find all occurrences

positions = find_substring_occurrences(sample_text, pattern)

# Print results

print(f"'{pattern}' occurrences:")

for pos in positions:

    print(f"Found at index {pos}")

# Count occurrences

print(f"Total occurrences: {len(positions)}")
```

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

Ans : from datetime import datetime

```
def convert_date_format(date_str):

    # Define the input date format and the desired output format

    input_format = '%Y-%m-%d'

    output_format = '%d-%m-%Y'

        # Parse the input date string into a datetime object

    date_obj = datetime.strptime(date_str, input_format)
    # Convert the datetime object to the desired string format

    new_date_str = date_obj.strftime(output_format)

        return new_date_str

# Test cases

dates = [

    '2024-08-21',

    '1999-12-31',

    '2000-01-01',

    '2010-05-15'

]

# Convert and print each date

for date in dates:

    converted_date = convert_date_format(date)

    print(f"Original date: {date} => Converted date: {converted_date}")
```

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']

```
Ans : import re

def find_decimal_numbers(text):

    # Compile the regular expression pattern

    pattern = re.compile(r'\b\d*\.\d{1,2}\b')

        # Find all matches in the text

    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 and print all decimal numbers with precision of 1 or 2

result = find_decimal_numbers(sample_text)

print(result)
```

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

```
Ans : import re

def find_numbers_and_positions(text):

    # Compile the regular expression pattern to find numbers (including decimals)

    pattern = re.compile(r'\b\d*\.?\d+\b')
    # Find all matches with their positions
    matches = pattern.finditer(text)
        results = []
    for match in matches:
        number = match.group()  # Extract the number
        position = match.start()  # Get the starting position of the number
        results.append((number, position))
        return results
# Sample text
sample_text = "The price is 123.45 and the discount is 50.25. Total 175.70."
# Find numbers and their positions
numbers_with_positions = find_numbers_and_positions(sample_text)
# Print results
print("Numbers and their positions:")
for number, position in numbers_with_positions:
    print(f"Number: {number}, Position: {position}")
```

Question 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

Ans : import re

```
def find_max_numeric_value(text):

    # Define the regex pattern to match numeric values

    pattern = re.compile(r'\b\d+\b')

        # Find all matches in the text

    matches = pattern.findall(text)

        # Convert matches to integers

    numbers = [int(match) for match in matches]

        # Return the maximum number

    if numbers:

        return max(numbers)

    else:

        return None

# Sample text

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

# Find the maximum numeric value

max_value = find_max_numeric_value(sample_text)



# Print the result
```

```
if max_value is not None:

    print(f"The maximum numeric value is: {
```

Question 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

Ans : import re

```
def insert_spaces(text):

    # Define the regex pattern to find capital letters that are preceded by lowercase letters or start of the
string

    pattern = re.compile(r'(?<!^)(?<!\s)(?<![A-Z])(?=[A-Z])')

        # Use the pattern to insert spaces before each capital letter

    spaced_text = pattern.sub(' ', text)

        return spaced_text

# Sample text

sample_text = "RegularExpressionIsAnImportantTopicInPython"

# Insert spaces and print the result

result = insert_spaces(sample_text)

print(result)
```

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

Ans : import re

```
def find_upper_lower_sequences(text):

    # Define the regex pattern for uppercase letter followed by lowercase letters

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

        # Find all matches in the text

    matches = pattern.findall(text)

        return matches

# Sample text

sample_text = "Python is an Amazing Programming Language for Data Science."

# Find and print all sequences of one uppercase letter followed by lowercase letters

sequences = find_upper_lower_sequences(sample_text)

print("Sequences of one uppercase letter followed by lowercase letters:")

print(sequences)
```

Question 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

```
Ans : import re

def remove_continuous_duplicates(text):

    # Define the regex pattern to match consecutive duplicate words

    pattern = re.compile(r'\b(\w+)\s+\1\b', re.IGNORECASE)

        # Use re.sub to replace consecutive duplicate words with a single occurrence

    result = pattern.sub(r'\1', text)

        return result

# Sample text

sample_text = "Hello hello world world"

# Remove continuous duplicate words and print the result

result = remove_continuous_duplicates(sample_text)

print(result)
```

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

```
Ans : import re

def ends_with_alphanumeric(text):

    # Define the regex pattern to check if the string ends with an alphanumeric character

    pattern = re.compile(r'\w$')

        # Use pattern.match to check if the text ends with an alphanumeric character

    if pattern.search(text):
        return True
    else:
        return False
# Test cases
test_strings = [
    "Hello World1",     # Ends with alphanumeric character '1'
    "Python@3",         # Ends with alphanumeric character '3'
    "Example text!",    # Ends with non-alphanumeric character '!'
    "JustText",         # Ends with alphanumeric character 't'
    "End with space ",  # Ends with space (not alphanumeric)
]
# Check each string and print if it ends with an alphanumeric character
for text in test_strings:
    if ends_with_alphanumeric(text):
        print(f"'{text}' ends with an alphanumeric character.")
    else:
        print(f"'{text}' does not end with an alphanumeric character.")
```

Question 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 <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo"""

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

Ans : import re

```
def extract_hashtags(text):

    # Define the regex pattern to match 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 #Demonetization as the same has rendered USELESS <ed><U+00A0><U+00BD><ed><U+00B1><U+0089> "acquired funds" No wo"""

# Extract hashtags and print the result

hashtags = extract_hashtags(sample_text)

print(hashtags)
```

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??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are protesting #demonetization are all different party leaders"

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

Ans : import re

```
def remove_unicode_symbols(text):

    # Define the regex pattern to match <U+...> like symbols

    pattern = re.compile(r'<U\+[0-9A-Fa-f]{4,}>')

        # Replace the matched patterns with an empty string

    cleaned_text = pattern.sub('', text)

        return cleaned_text

# Sample text

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"

# Remove <U+...> symbols and print the result

cleaned_text = remove_unicode_symbols(sample_text)

print(cleaned_text)
```

Question 29- Write a python program to extract dates from the text stored in the text file.

Sample Text: Ron was born on 12-09-1992 and he was admitted to school 15-12-1999.

Note- Store this sample text in the file and then extract dates.

Ans : import re

```
def extract_dates_from_file(file_path):
    # Define the regex pattern to match dates in dd-mm-yyyy format
    date_pattern = re.compile(r'\b\d{2}-\d{2}-\d{4}\b')
        with open(file_path, 'r') as file:
        text = file.read()
        # Find all matches in the text
    dates = date_pattern.findall(text)
        return dates
# Path to the text file
file_path = 'sample_text.txt'
# Store sample text in the file
with open(file_path, 'w') as file:
    file.write('Ron was born on 12-09-1992 and he was admitted to school 15-12-1999.')
# Extract dates and print the result
extracted_dates = extract_dates_from_file(file_path)
print("Extracted dates:", extracted_dates)
```

Question 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.

Ans : import re

```
def remove_short_words(text):

    # Define the regex pattern to match words of length between 2 and 4

    pattern = re.compile(r'\b\w{2,4}\b')

        # Replace matches with an empty string

    cleaned_text = pattern.sub('', text).strip()

        # Optional: Normalize spaces to ensure no extra spaces are left

    cleaned_text = re.sub(r'\s+', ' ', cleaned_text)

        return cleaned_text

# Sample text

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."

# Remove short words and print the result

result = remove_short_words(sample_text)

print(result)
```