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:

Solx:- def replace_chars(text):

```
    chars_to_replace = [' ', ',', '.']
    for char in chars_to_replace:
        text = text.replace(char, ':')
    return text
sample_text = 'Python Exercises, PHP exercises.'
result = replace_chars(sample_text)
print(result)
```

expected output-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

Solution:- import pandas as pd

import re

Define the dictionary

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

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


# Define a function to remove unwanted characters
def clean_text(text):
    cleaned_text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove non-alphabetic characters except spaces
 return cleaned_text.strip()  # Remove leading and trailing spaces


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


# Output the cleaned DataFrame
print(df)
 SUMMARY
0  hello world
1   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.

Solution:- import re

```
def find_long_words(text):
    # Compile a regular expression pattern to match words with at least 4 characters
    pattern = re.compile(r'\b\w{4,}\b')
    # Find all matches of the pattern in the text
    long_words = pattern.findall(text)


    return long_words
```

```
# Test the function

text = "This is a sample sentence with words of different lengths, like apple, banana, cat, dog, elephant, and giraffe."

result = find_long_words(text)

print(result)
```

it finds all matches of this pattern in the input text using **pattern.findall(text)** and returns the list of long words found.

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.

Solx:-

```
import re

def find_words(string):
    pattern = re.compile(r'\b\w{3,5}\b')
    words = pattern.findall(string)
    return words

# Example usage:

text = "This is a sample string with words of varying lengths like cat, dog, and bird."

result = find_words(text)

print(result)
```

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

Solx:-

```
import re
```

```
def remove_parentheses(strings):
    pattern = re.compile(r'\([^)]*\)')  # matches parentheses and their contents
    result = []
    for s in strings:
        cleaned = re.sub(pattern, '', s)  # remove parentheses and their contents
        result.append(cleaned)
    return result


# Sample usage
text = ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data Science World)", "Data (Scientist)"]
cleaned_text = remove_parentheses(text)
print(cleaned_text)
```

output:-

['example ', 'hr@fliprobo ', 'github ', 'Hello Data Science World', 'Data Scientist']

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.

Solx:- import re

```
# Open the input file and read the contents
with open('input.txt', 'r') as file:
    text = file.read()

# Convert the text to a list of strings
strings = eval(text)
```

```
# Define the regular expression pattern
pattern = re.compile(r'\([^)]*\)')

# Remove parentheses and their contents from each string
cleaned_strings = [re.sub(pattern, '', s) for s in strings]

# Convert the cleaned list of strings back to a string representation
cleaned_text = str(cleaned_strings)

# Write the cleaned text to an output file
with open('output.txt', 'w') as file:
    file.write(cleaned_text)

print("Cleaned text has been written to 'output.txt'")
```

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’]

Solx:-

```
import re

def split_by_uppercase(text):
    pattern = r'(?=[A-Z])'  # Matches positions before uppercase letters
    return re.split(pattern, text)

# Sample usage
sample_text = "ImportanceOfRegularExpressionsInPython"
result = split_by_uppercase(sample_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

Solx:- import re

```
def insert_spaces(text):
    pattern = r'(?<=\D)(?=\d)'  # Matches positions between non-digit and digit
    return re.sub(pattern, ' ', text)

# Sample usage
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
result = insert_spaces(sample_text)
print(result)
```

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

Solx:- import re

```
def insert_spaces(text):
    pattern = r'(?<=\D)(?=\d|\w*[A-Z])'  # Matches positions between non-digit and digit/uppercase word
    return re.sub(pattern, ' ', text)

# Sample usage
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
result = insert_spaces(sample_text)
print(result)
```

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

Solx:- import pandas as pd

```
# Read data into a string

data = """Country,Region,Happiness Rank,Happiness Score,Standard Error,Economy (GDP per Capita),Family,Health (Life Expectancy),Freedom,Trust (Government Corruption),Generosity,Dystopia Residual

Switzerland,Western Europe,1,7.587,0.03411,1.39651,1.34951,0.94143,0.66557,0.41978,0.29678,2.51738

Iceland,Western Europe,2,7.561,0.04884,1.30232,1.40223,0.94784,0.62877,0.14145,0.4363,2.70201

...

"""


# Create DataFrame from string

df = pd.read_csv(pd.compat.StringIO(data))


# Add new column 'first_five_letters'

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


print(df.head())
```

output:-

	Country	Region	Happiness Rank	Happiness Score	...	Generosity	Dystopia Residual	first_five_letters
0	Switzerland	Western Europe	1	7.587	...	0.29678	2.51738	Switz
1	Iceland	Western Europe	2	7.561	...	0.43630	2.70201	Icelan

2	Denmark	Western Europe	3	7.527	...	0.34139	2.49204	Denmar
3	Norway	Western Europe	4	7.522	...	0.34699	2.46531	Norway
4	Canada	North America	5	7.427	...	0.45811	2.45176	Canada

[5 rows x 13 columns]

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

Solx:- import re

```
def is_valid_string(string):
    pattern = r'^[a-zA-Z0-9_]+$'
    if re.match(pattern, string):
        return True
    else:
        return False

# Test the function
print(is_valid_string("Hello_World123"))    # True
print(is_valid_string("Python@3.9"))       # False (contains a special character '@')
print(is_valid_string("123_abc"))          # True
print(is_valid_string("foo bar"))          # False (contains a space)
```

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

Solx:-
```
def starts_with_number(string, num):
    """
    Checks if a given string starts with a specific number.

    Args:
        string (str): The input string.
        num (int): The number to check for at the start of the string.
```

```
    Returns:
        bool: True if the string starts with the specified number, False otherwise.
    """
    num_str = str(num)  # Convert the number to a string
    if string.startswith(num_str):
        return True
    else:
        return False


# Example usage
print(starts_with_number("123hello", 123))  # True
print(starts_with_number("456world", 123))  # False
print(starts_with_number("7890abc", 78))    # True
print(starts_with_number("0912xyz", 912))   # False
```

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

Solx:-

```
def remove_leading_zeros(ip_address):
    """
    Removes leading zeros from each octet in an IP address.

    Args:
        ip_address (str): The input IP address.

    Returns:
        str: The IP address with leading zeros removed from each octet.
    """
    # 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 = '.'.join(cleaned_octets)

    return cleaned_ip

# Example usage
print(remove_leading_zeros('192.168.001.003'))  # Output: 192.168.1.3
print(remove_leading_zeros('10.020.030.040'))   # Output: 10.20.30.40
print(remove_leading_zeros('172.16.0.1'))       # Output: 172.16.0.1
```

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.

Solx:- import re

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

    # Use regular expression to find the date string
    date_pattern = r"(\w+\s\d+\w+\s\d+)"
    match = re.search(date_pattern, text)

    if match:
```

```
        date_string = match.group(1)
        return date_string
    else:
        return "Date string not found."

# Example usage
file_path = "sample.txt"
extracted_date = extract_date(file_path)
print(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'

Solx:-

```
def search_strings(text, search_words):
    for word in search_words:
        if word in text:
            print(f"'{word}' found in the text.")
        else:
            print(f"'{word}' not found in the text.")

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

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

search_strings(sample_text, search_words)
```

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'

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

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

Pattern : 'exercises'.

Solx:- def find_substrings(text, pattern):

```
    start = 0
    substrings = []

    while True:
        start = text.find(pattern, start)
        if start == -1:
            break
        substrings.append(text[start:start+len(pattern)])
        start += len(pattern)

    return substrings

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

# Pattern to search
pattern = 'exercises'

# Find substrings
substrings = find_substrings(sample_text, pattern)
```

```
# Print results
print(f"Substrings found: {substrings}")
```

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

Solx:-
```
def find_substring_occurrences(text, substring):
    occurrences = []
    start = 0
    text_length = len(text)
    substring_length = len(substring)

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

    return occurrences

# Example usage
sample_text = "Python exercises, PHP exercises, C# exercises"
substring = "exercises"

occurrences = find_substring_occurrences(sample_text, substring)

for substring, position in occurrences:
    print(f"Found '{substring}' at position {position}")
```

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

Solx:- from datetime import datetime

```
def convert_date_format(date_str):
    # Convert the string to a datetime object
    date_obj = datetime.strptime(date_str, "%Y-%m-%d")

    # Convert the datetime object to the desired format
    new_date_str = date_obj.strftime("%d-%m-%Y")

    return new_date_str

# Example usage
original_date = "2023-04-15"
converted_date = convert_date_format(original_date)
print(f"Original date: {original_date}")
print(f"Converted date: {converted_date}")
```

output:- Original date: 2023-04-15

Converted date: 15-04-2023

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']

Solx:- import re

```
def find_decimals(text):
    pattern = r'\b\d+\.\d{1,2}\b'  # Regular expression pattern
    regex = re.compile(pattern)
    return regex.findall(text)
```

```
# Sample usage
sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"
decimals = find_decimals(sample_text)
print(decimals)
```

output:- ['01.12', '145.8', '3.01', '27.25', '0.25']

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

Solx:-

```
import re

def find_numbers_and_positions(text):
    pattern = r'\d+'  # Regular expression pattern to match numbers
    numbers = re.findall(pattern, text)
    positions = []

    for match in re.finditer(pattern, text):
        positions.append((match.group(), match.start(), match.end()))

    return numbers, positions

# Example usage
sample_string = "There are 3 apples and 5 oranges in the basket. I have 2 books."

numbers, positions = find_numbers_and_positions(sample_string)

print("Numbers found:", numbers)
print("Positions:")
for num, start, end in positions:
    print(f"Number: {num}, Start: {start}, End: {end}")
```

output:- Numbers found: ['3', '5', '2']

Positions:

Number: 3, Start: 11, End: 12

Number: 5, Start: 19, End: 20

Number: 2, Start: 48, End: 49

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

Solx:- import re

```
def extract_max_number(text):
    # Define the regular expression pattern
    pattern = r'\d+'  # Match one or more digits

    # Find all numeric values in the string
    numbers = re.findall(pattern, text)

    # Convert the numeric strings to integers
    numbers = [int(num) for num in numbers]

    # Find the maximum value
    if numbers:
        max_number = max(numbers)
        return max_number
    else:
        return None

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

```
# Extract the maximum numeric value
max_value = extract_max_number(sample_text)
if max_value:
    print(f"The maximum numeric value is: {max_value}")
else:
    print("No numeric values found in the string.")
```

output:- The maximum numeric value is: 950

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

Solx:-

```
import re

def insert_spaces(text):
    pattern = r'(?<!^)(?=[A-Z])'
    new_text = re.sub(pattern, ' ', text)
    return new_text

# Sample text
sample_text = "RegularExpressionIsAnImportantTopicInPython"

# Insert spaces between words starting with capital letters
formatted_text = insert_spaces(sample_text)
print(formatted_text)
```

output:- Regular Expression Is An Important Topic In Python

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

Solx:-

```
import re
```

```
pattern = r'[A-Z][a-z]+'


test_strings = [
    "HelloWorld",
    "PythonRegex",
    "JavaScript",
    "123ABC456def",
    "A",
    "Aa",
    "AAa",
    "AaBb"
]


for test_string in test_strings:
    matches = re.findall(pattern, test_string)
    if matches:
        print(f"Matches in '{test_string}': {', '.join(matches)}")
    else:
        print(f"No matches in '{test_string}'")
```

Matches in 'HelloWorld': Hello

Matches in 'PythonRegex': Python

No matches in 'JavaScript'

Matches in '123ABC456def': Abc, Def

No matches in 'A'

Matches in 'Aa': Aa

No matches in 'AAa'

Matches in 'AaBb': Aa, Bb

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

Solx:- import re

```
def remove_continuous_duplicates(sentence):
    pattern = r'\b(\w+)(\s+\1\b)+'
    return re.sub(pattern, r'\1', sentence)

# Sample text
sample_text = "Hello hello world world"

# Remove continuous duplicate words
result = remove_continuous_duplicates(sample_text)
print("Original Sentence:", sample_text)
print("After Removing Continuous Duplicates:", result)
```

Original Sentence: Hello hello world world

After Removing Continuous Duplicates: Hello world

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

Solx:- import re

```
def is_valid_string(input_string):
    pattern = r'^.*[a-zA-Z0-9]$'
    if re.match(pattern, input_string):
        print(f"'{input_string}' is a valid string ending with an alphanumeric character.")
    else:
        print(f"'{input_string}' is not a valid string ending with an alphanumeric character.")

# Test cases
```

```
is_valid_string("hello123")  # Valid

is_valid_string("python123@")  # Invalid

is_valid_string("123python")  # Valid

is_valid_string("hello world")  # Invalid

is_valid_string("python123")  # Valid
```

output:- 'hello123' is a valid string ending with an alphanumeric character.

'python123@' is not a valid string ending with an alphanumeric character.

'123python' is a valid string ending with an alphanumeric character.

'hello world' is not a valid string ending with an alphanumeric character.

'python123' is a valid string ending 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']

```
import re


def extract_hashtags(text):
    pattern = r'#\w+'
    hashtags = re.findall(pattern, 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
hashtags = extract_hashtags(sample_text)
print("Extracted Hashtags:")
print(hashtags)
```

output :- 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??<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

Solx:- import re

```
def remove_unicode_symbols(text):
    pattern = re.compile(r'<U\+[0-9A-Fa-f]{4}>')
    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"

# Call the function
output = remove_unicode_symbols(sample_text)

# Print the output
print(output)
```

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.

Solx:- import re

```
def extract_dates_from_file(file_path):
    dates = []
    with open(file_path, 'r') as file:
        text = file.read()
        pattern = re.compile(r'\b\d{2}-\d{2}-\d{4}\b')
        dates = pattern.findall(text)
    return dates

# Path to the text file
file_path = 'sample_text.txt'

# Call the function to extract dates
extracted_dates = extract_dates_from_file(file_path)

# Print the extracted dates
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.

Solx:- import re

```
def remove_words(string):
```

```
    pattern = re.compile(r'\b\w{2,4}\b')
    cleaned_string = pattern.sub('', string)
    return cleaned_string

# 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."

# Call the function
output = remove_words(sample_text)

# Print the output
print(output)
```