**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: def replace\_chars(text):

replacements = {' ': ':', ',': ':', '.': ':'}

for old\_char, new\_char in replacements.items():

text = text.replace(old\_char, new\_char)

return text

sample\_text = 'Python Exercises, PHP exercises.'

result = replace\_chars(sample\_text)

print(result)

**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

# 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 remove\_special\_chars(text):

# Use regular expression to remove non-word characters

return re.sub(r'[^\w\s]', '', text)

# Apply the function to the 'SUMMARY' column

df['SUMMARY'] = df['SUMMARY'].apply(remove\_special\_chars)

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):

# Compile a regular expression pattern to match words of at least 4 characters

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

# Use findall() method to find all matching words

long\_words = pattern.findall(text)

return long\_words

# Example usage:

text = "This is a sample sentence with long words like elephant, giraffe, and pythonic."

result = find\_long\_words(text)

print(result)

**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\_words(text):

# Compile a regular expression pattern to match words of three, four, or five characters

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

# Use findall() method to find all matching words

words = pattern.findall(text)

return words

# Example usage:

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

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

Ans: import re

def remove\_parenthesis(strings):

# Compile a regular expression pattern to match parentheses

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

# Use sub() method to replace parentheses with an empty string

cleaned\_strings = [pattern.sub('', string) for string in strings]

return cleaned\_strings

# Example usage:

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

result = remove\_parenthesis(sample\_text)

for string in result:

print(string)

**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\_parenthesis(text):

# Compile a regular expression pattern to match parentheses and their contents

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

# Use sub() method to replace the matched pattern with an empty string

cleaned\_text = pattern.sub('', text)

return cleaned\_text

# Read the sample text from the file

file\_path = 'sample\_text.txt'

with open(file\_path, 'r') as file:

sample\_text = file.read()

# Remove parenthesis area from the text

cleaned\_text = remove\_parenthesis(sample\_text)

# Write the cleaned text to a new file

output\_file\_path = 'cleaned\_text.txt'

with open(output\_file\_path, 'w') as output\_file:

output\_file.write(cleaned\_text)

print("Text with parenthesis removed has been saved to cleaned\_text.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’]

Ans: import re

sample\_text = "ImportanceOfRegularExpressionsInPython"

split\_words = re.findall('[A-Z][^A-Z]\*', sample\_text)

print(split\_words)

Output: ['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']

**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):

# Use regular expression to find the boundary between a digit and a non-digit character

# Insert a space at that boundary

modified\_text = re.sub(r'(\d)(?=[A-Za-z])', r'\1 ', text)

return modified\_text

# Sample Text

sample\_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Output

output\_text = insert\_spaces(sample\_text)

print(output\_text)

Output: RegularExpression 1IsAn 2ImportantTopic 3InPython

**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):

# Use regular expression to find the boundary between a capital letter or a digit

# and a lowercase letter

# Insert a space at that boundary

modified\_text = re.sub(r'([A-Z\d])(?=[a-z])', r'\1 ', text)

return modified\_text

# Sample Text

sample\_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Output

output\_text = insert\_spaces(sample\_text)

print(output\_text)

Output: RegularExpression 1 IsAn 2 ImportantTopic 3 InPython

**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

# Read the data from the provided GitHub link

url = "https://raw.githubusercontent.com/dsrscientist/DSData/master/happiness\_score\_dataset.csv"

df = pd.read\_csv(url)

# Extract the first 6 letters of each country

df['first\_five\_letters'] = df['Country'].str[:6]

# Display the dataframe

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(text):

# Define a regular expression pattern

pattern = re.compile(r'^[a-zA-Z0-9\_]+$')

# Use match() method to check if the string matches the pattern

if pattern.match(text):

return True

else:

return False

# Test the function

test\_strings = ["Hello\_World123", "abcDEF\_456", "123", "abc@123", "abc 123"]

for string in test\_strings:

if match\_string(string):

print(f"'{string}' matches the pattern.")

else:

print(f"'{string}' does not match the pattern.")

Output: 'Hello\_World123' matches the pattern.

'abcDEF\_456' matches the pattern.

'123' matches the pattern.

'abc@123' does not match the pattern.

'abc 123' does not match the pattern.

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

Ans: def starts\_with\_number(string, number):

# Convert the number to a string

number\_str = str(number)

# Check if the string starts with the specified number

if string.startswith(number\_str):

return True

else:

return False

# Test the function

test\_strings = ["123abc", "456def", "789ghi", "abc123", "def456"]

specific\_number = 123

for string in test\_strings:

if starts\_with\_number(string, specific\_number):

print(f"'{string}' starts with the number {specific\_number}.")

else:

print(f"'{string}' does not start with the number {specific\_number}.")

output: '123abc' starts with the number 123.

'456def' starts with the number 123.

'789ghi' does not start with the number 123.

'abc123' does not start with the number 123.

'def456' does not start with the number 123.

**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 components (octets)

octets = ip\_address.split('.')

# Convert each octet to an integer and then back to a string

octets = [str(int(octet)) for octet in octets]

# Join the octets with '.'

cleaned\_ip = '.'.join(octets)

return cleaned\_ip

# Test the function

ip\_address = "192.168.001.001"

cleaned\_ip = remove\_leading\_zeros(ip\_address)

print("Original IP Address:", ip\_address)

print("Cleaned IP Address:", cleaned\_ip)

Output: Original IP Address: 192.168.001.001

Cleaned IP Address: 192.168.1.1

**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.

Ans: import re

def extract\_date\_from\_text(text):

# Define the regular expression pattern

pattern = re.compile(r'\b(?:January|February|March|April|May|June|July|August|September|October|November|December)\s+\d{1,2}(?:st|nd|rd|th)?\s+\d{4}\b')

# Use findall() to extract all matching date strings

dates = pattern.findall(text)

return dates

# Read the sample text from the file

file\_path = 'sample\_text.txt'

with open(file\_path, 'r') as file:

sample\_text = file.read()

# Extract the date string

dates = extract\_date\_from\_text(sample\_text)

# Print the extracted date string(s)

print("Extracted Date String(s):")

for date in dates:

print(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\_words(text, words):

for word in words:

if text.find(word) != -1:

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.'

# Searched words

searched\_words = ['fox', 'dog', 'horse']

# Search for words

search\_words(sample\_text, searched\_words)

Output: 'fox' found in the text.

'dog' found in the text.

'horse' not found 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\_word(text, word):

index = text.find(word)

if index != -1:

print(f"'{word}' found at index {index} 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.'

# Searched word

searched\_word = 'fox'

# Search for the word

search\_word(sample\_text, searched\_word)

output: 'fox' found at index 16 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: def find\_substrings(text, pattern):

start = 0

occurrences = []

while True:

index = text.find(pattern, start)

if index == -1:

break

occurrences.append(index)

start = index + 1

return occurrences

# Sample text

sample\_text = 'Python exercises, PHP exercises, C# exercises'

# Pattern to search

pattern = 'exercises'

# Find substrings

occurrences = find\_substrings(sample\_text, pattern)

# Print the occurrences

if occurrences:

print(f"The pattern '{pattern}' is found at the following indices:")

for index in occurrences:

print(index)

else:

print(f"The pattern '{pattern}' is not found in the text.")

output: The pattern 'exercises' is found at the following indices:

7

22

38

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

Ans: def find\_occurrences(text, pattern):

occurrences = []

start = 0

while True:

index = text.find(pattern, start)

if index == -1:

break

occurrences.append((pattern, index))

start = index + len(pattern)

return occurrences

# Sample text

sample\_text = 'Python exercises, PHP exercises, C# exercises'

# Pattern to search

pattern = 'exercises'

# Find occurrences and positions of substrings

occurrences = find\_occurrences(sample\_text, pattern)

# Print the occurrences and positions

if occurrences:

print(f"The pattern '{pattern}' occurs at the following positions:")

for occurrence in occurrences:

print(f"Position: {occurrence[1]}, Occurrence: {occurrence[0]}")

else:

print(f"The pattern '{pattern}' is not found in the text.")

Output: The pattern 'exercises' occurs at the following positions:

Position: 7, Occurrence: exercises

Position: 22, Occurrence: exercises

Position: 38, Occurrence: exercises

**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(date\_str):

# Parse the date string

date\_obj = datetime.strptime(date\_str, '%Y-%m-%d')

# Format the date object in dd-mm-yyyy format

formatted\_date = date\_obj.strftime('%d-%m-%Y')

return formatted\_date

# Example usage:

date\_str = '2022-04-03'

converted\_date = convert\_date(date\_str)

print("Original date (yyyy-mm-dd):", date\_str)

print("Converted date (dd-mm-yyyy):", converted\_date)

Output: Original date (yyyy-mm-dd): 2022-04-03

Converted date (dd-mm-yyyy): 03-04-2022

**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 a regular expression pattern to match decimal numbers with precision of 1 or 2

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

# Use findall() method to find all matching decimal numbers

decimal\_numbers = pattern.findall(text)

return decimal\_numbers

# Sample text

sample\_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

# Find decimal numbers

result = find\_decimal\_numbers(sample\_text)

print("Decimal numbers with precision of 1 or 2:", result)

Output: Decimal numbers with precision of 1 or 2: ['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.

Ans: def separate\_numbers(text):

numbers = []

positions = []

for index, char in enumerate(text):

if char.isdigit():

numbers.append(char)

positions.append(index)

return numbers, positions

# Sample text

sample\_text = "abc123def456ghi789"

# Separate numbers and positions

numbers, positions = separate\_numbers(sample\_text)

# Print numbers and their positions

print("Numbers:", numbers)

print("Positions:", positions)

Output: Numbers: ['1', '2', '3', '4', '5', '6', '7', '8', '9']

Positions: [3, 4, 5, 9, 10, 11, 15, 16, 17]

**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 extract\_max\_numeric\_value(text):

# Find all numeric values in the string

numeric\_values = re.findall(r'\d+', text)

# Convert the numeric values to integers and find the maximum

max\_value = max(map(int, numeric\_values))

return max\_value

# Sample text

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

# Extract maximum numeric value

max\_numeric\_value = extract\_max\_numeric\_value(sample\_text)

# Print the result

print("Maximum numeric value:", max\_numeric\_value)

Output: Maximum numeric value: 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

Ans: import re

def insert\_spaces(text):

# Use regular expression to find the boundary between lowercase and uppercase letters

# Insert a space at that boundary

modified\_text = re.sub(r'(?<=[a-z])(?=[A-Z])', ' ', text)

return modified\_text

# Sample text

sample\_text = "RegularExpressionIsAnImportantTopicInPython"

# Insert spaces between words starting with capital letters

output\_text = insert\_spaces(sample\_text)

# Print the result

print("Original text:", sample\_text)

print("Modified text:", output\_text)

output: Original text: RegularExpressionIsAnImportantTopicInPython

Modified text: 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

Ans: import re

def find\_sequences(text):

# Define the regular expression pattern

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

# Use findall() to find all matching sequences

sequences = pattern.findall(text)

return sequences

# Sample text

sample\_text = "The quick Brown fox Jumps over the Lazy dog"

# Find sequences of one uppercase letter followed by lowercase letters

result = find\_sequences(sample\_text)

# Print the result

print("Sequences found:", result)

Output: Sequences found: ['Brown', 'Jumps', 'Lazy']

**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\_duplicate\_words(sentence):

# Use regular expression to remove continuous duplicate words

cleaned\_sentence = re.sub(r'\b(\w+)(\s+\1\b)+', r'\1', sentence, flags=re.IGNORECASE)

return cleaned\_sentence

# Sample text

sample\_text = "Hello hello world world"

# Remove continuous duplicate words

cleaned\_text = remove\_duplicate\_words(sample\_text)

# Print the result

print("Original sentence:", sample\_text)

print("Cleaned sentence:", cleaned\_text)

Output: Original sentence: Hello hello world world

Cleaned sentence: Hello hello world

**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 regular expression pattern

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

# Use search() to find if the text ends with an alphanumeric character

match = pattern.search(text)

# Return True if match is found, otherwise False

return bool(match)

# Test strings

test\_strings = ["Hello123", "world!", "12345", "python3", "ends-with-alphanumeric"]

# Check if each string ends with an alphanumeric character

for string in test\_strings:

if ends\_with\_alphanumeric(string):

print(f"'{string}' ends with an alphanumeric character.")

else:

print(f"'{string}' does not end with an alphanumeric character.")

Output: 'Hello123' ends with an alphanumeric character.

'world!' does not end with an alphanumeric character.

'12345' ends with an alphanumeric character.

'python3' ends with an alphanumeric character.

'ends-with-alphanumeric' 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 regular expression pattern to match hashtags

pattern = re.compile(r'#\w+')

# Use findall() to extract all matching hashtags

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

hashtags = extract\_hashtags(sample\_text)

# Print the extracted hashtags

print("Extracted hashtags:", 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

Ans: import re

def remove\_special\_symbols(text):

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

pattern = re.compile(r'<U\+[A-F0-9]{4}>')

# Use sub() to replace all occurrences of the pattern 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+..> like symbols

cleaned\_text = remove\_special\_symbols(sample\_text)

# Print the cleaned text

print("Cleaned text:", cleaned\_text)

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

**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\_text\_file(file\_path):

# Read the text from the file

with open(file\_path, 'r') as file:

text = file.read()

# Define the regular expression pattern to match dates in dd-mm-yyyy format

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

# Use findall() to extract all matching dates

dates = pattern.findall(text)

return dates

# File path where the sample text is stored

file\_path = 'sample\_text.txt'

# Extract dates from the text file

extracted\_dates = extract\_dates\_from\_text\_file(file\_path)

# Print the extracted dates

print("Extracted dates:", extracted\_dates)

Output: Extracted dates: ['12-09-1992', '15-12-1999']

**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\_words\_of\_length\_2\_to\_4(text):

# Define the regular expression pattern to match words of length 2 to 4

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

# Use sub() to replace all occurrences of the pattern with an empty string

cleaned\_text = pattern.sub('', 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 words of length between 2 and 4

cleaned\_text = remove\_words\_of\_length\_2\_to\_4(sample\_text)

# Print the cleaned text

print("Cleaned text:", cleaned\_text)

output: Cleaned text: following example creates ArrayList capacity elements. 4 elements then added ArrayList ArrayList trimmed accordingly.