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:

Solution:- import re

```
def replace_with_colon(text):
    pattern = r'[ ,.]'
    replaced_text = re.sub(pattern, ':', text)
    return replaced_text

sample_text = 'Python Exercises, PHP exercises.'
modified_text = replace_with_colon(sample_text)
print("Original Text:", sample_text)
print("Modified Text:", modified_text)
```

```
o/p:-- Original Text: Python Exercises, PHP exercises.
Modified Text: 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


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


# Create DataFrame
df = pd.DataFrame(data)


# Remove unwanted characters from the 'SUMMARY' column using regular expressions
df['SUMMARY'] = df['SUMMARY'].apply(lambda x: re.sub(r'[^\w\s]', '', x))


# Display the cleaned DataFrame
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.

Solution:--

```
    import re


def find_words_at_least_four_chars(text):
   pattern = re.compile(r'\b\w{4,}\b')  # Regex pattern to find words with at least 4 characters
   matches = pattern.findall(text)
   return matches


# Example usage:
input_text = "This is a sample text to find words with at least four characters like hello, world, python, example"
```

result = find_words_at_least_four_chars(input_text)

print("Words with at least four characters:", result)

```
Words with at least four characters: ['This', 'sample', 'text', 'find', 'w
ords', 'with', 'least', 'four', 'characters', 'like', 'hello', 'world', 'p
ython', 'example']
```

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.

Solution:- import re

```
def find_words_three_to_five_chars(text):
    pattern = re.compile(r'\b\w{3,5}\b')  # Regex pattern to find words with 3, 4, or 5 characters
    matches = pattern.findall(text)
    return matches

# Example usage:
input_text = "This is a sample text to find words with three, four, or five characters like hello, world, python, example"
result = find_words_three_to_five_chars(input_text)
print("Words with three, four, or five characters:", result)
```

```
Words with three, four, or five characters: ['This', 'text', 'find', 'word
s', 'with', 'three', 'four', 'five', 'like', 'hello', 'world']
```

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

Solution:-- import re

```
def remove_parentheses(strings):
    pattern = re.compile(r'\(|\)')  # Regex pattern to match parentheses
    modified_strings = [pattern.sub('', s) for s in strings]
    return modified_strings

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

result = remove_parentheses(sample_text)
print("Strings after removing parentheses:")
for string in result:
    print(string)
```

```
Strings after removing parentheses:
example .com
hr@fliprobo .com
github .com
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.

Solution:-- ["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data Science World)", "Data (Scientist)"]

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']

Solution:-- import re

```
sample_text = "ImportanceOfRegularExpressionsInPython"

# Split the string into uppercase letters using regex
uppercase_letters = re.findall('[A-Z][^A-Z]*', sample_text)

print("Uppercase letters:", uppercase_letters)
```

```
Uppercase letters: ['Importance', 'Of', 'Regular', 'Expressions', 'In', 'P
ython']
```

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

Sample Text: "RegularExpression1IsAn2ImportantTopic3InPython"

Expected Output: RegularExpression 1IsAn 2ImportantTopic 3InPython

Solution:-- import re

```
def insert_spaces(text):
    modified_text = re.sub(r'(?<=[0-9])(?=[A-Za-z])', ' ', text)
    return modified_text

sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
modified_text = insert_spaces(sample_text)
print("Original Text:", sample_text)
print("Modified Text:", modified_text)
```

```
Original Text: RegularExpression1IsAn2ImportantTopic3InPython
Modified Text: 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

Solution:-- import re

```
def insert_spaces(text):
    modified_text = re.sub(r'(?<=\d)(?=[A-Za-z])', ' ', text)
    return modified_text

# Sample Text
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Insert spaces between words starting with numbers
result = insert_spaces(sample_text)
print("Modified text with spaces:", result)
```

```
Modified text with spaces: RegularExpression1 IsAn2 ImportantTopic3 InPyth
on
```

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

Solution:-- mport pandas as pd

Raw data

raw_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

Denmark,Western Europe,3,7.527,0.03328,1.32548,1.36058,0.87464,0.64938,0.48357,0.34139,2.49204

Norway,Western Europe,4,7.522,0.0388,1.459,1.33095,0.88521,0.66973,0.36503,0.34699,2.46531

Canada,North America,5,7.427,0.03553,1.32629,1.32261,0.90563,0.63297,0.32957,0.45811,2.45176

Finland,Western Europe,6,7.406,0.0314,1.29025,1.31826,0.88911,0.64169,0.41372,0.23351,2.61955

Netherlands,Western Europe,7,7.378,0.02799,1.32944,1.28017,0.89284,0.61576,0.31814,0.4761,2.4657

Sweden,Western Europe,8,7.364,0.03157,1.33171,1.28907,0.91087,0.6598,0.43844,0.36262,2.37119

New Zealand,Australia and New Zealand,9,7.286,0.03371,1.25018,1.31967,0.90837,0.63938,0.42922,0.47501,2.26425

Australia,Australia and New Zealand,10,7.284,0.04083,1.33358,1.30923,0.93156,0.65124,0.35637,0.43562,2.26646

Israel,Middle East and Northern Africa,11,7.278,0.0347,1.22857,1.22393,0.91387,0.41319,0.07785,0.33172,3.08854

Costa Rica,Latin America and Caribbean,12,7.226,0.04454,0.95578,1.23788,0.86027,0.63376,0.10583,0.25497,3.17728

```
Austria,Western Europe,13,7.2,0.03751,1.33723,1.29704,0.89042,0.62433,0.18676,0.33088,2.5332

Mexico,Latin America and
Caribbean,14,7.187,0.04176,1.02054,0.91451,0.81444,0.48181,0.21312,0.14074,3.60214

United States,North
America,15,7.119,0.03839,1.39451,1.24711,0.86179,0.54604,0.1589,0.40105,2.51011

Brazil,Latin America and
Caribbean,16,6.983,0.04076,0.98124,1.23287,0.69702,0.49049,0.17521,0.14574,3.26001

'''


# Convert raw data to DataFrame

data = [line.split(',') for line in raw_data.split('\n')]

header = data[0]

rows = data[1:]


# Create DataFrame

df = pd.DataFrame(rows, columns=header)


# Extract first 6 letters of each country and store in a new column

df['first_five_letters'] = df['Country'].str[:5]


print(df[['Country', 'first_five_letters']])
```

```
          Country first_five_letters
0     Switzerland              Switz
1         Iceland              Icela
2         Denmark              Denma
3          Norway              Norwa
4          Canada              Canad
5         Finland              Finla
6     Netherlands              Nethe
7          Sweden              Swede
8     New Zealand              New Z
9       Australia              Austr
10         Israel              Israe
11     Costa Rica              Costa
12        Austria              Austr
```

```
13         Mexico              Mexic
14  United States              Unite
15         Brazil              Brazi
```

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

Solution:-- import re

```
def match_string(text):
    pattern = re.compile(r'^[a-zA-Z0-9_]*$')
    match = pattern.match(text)
    return bool(match)

# Sample strings to test
sample_strings = [
    "Hello_World123",
    "This_is_a_test",
    "Test_123!!!",
    "GoodMorning"
]

for string in sample_strings:
    result = match_string(string)
    print(f"String: {string}, Matched: {result}")
```

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

Solution:-- def starts_with_number(text, number):

```
    return text.startswith(str(number))

# Sample string and number to check
```

```
sample_string = "123ABC"
specific_number = 123

# Check if the string starts with the specific number
result = starts_with_number(sample_string, specific_number)
if result:
    print(f"The string '{sample_string}' starts with the number {specific_number}.")
else:
    print(f"The string '{sample_string}' does not start with the number {specific_number}.")
```

```
The string '123ABC' starts with the number 123.
```

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

Solution:--

```
def remove_leading_zeros(ip_address):
    # Split the IP address into individual octets and remove leading zeros
    octets = [str(int(octet)) for octet in ip_address.split(".")]
    return ".".join(octets)

# Sample IP address with leading zeros
ip_with_zeros = "192.168.001.001"

# Remove leading zeros from the IP address
ip_without_zeros = remove_leading_zeros(ip_with_zeros)

print("IP address with leading zeros:", ip_with_zeros)
print("IP address without leading zeros:", ip_without_zeros)
```

```
IP address with leading zeros: 192.168.001.001
IP address without leading zeros: 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.

Solution:-- import re

```
def find_dates_in_text(file_path):
    with open(file_path, 'r') as file:
        text = file.read()
        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')
        dates = pattern.findall(text)
        return dates

# Sample text file path
file_path = 'sample_text.txt'

# Find dates in the text file
date_matches = find_dates_in_text(file_path)
print("Date strings found in the text:")
for date in date_matches:
    print(' '.join(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'

Solution:-- def search_strings(text):

```
    strings_to_search = ["quick", "fox", "lazy"]  # List of strings to search for


    for string in strings_to_search:
        if string in text:
            print(f"'{string}' found in the text.")
        else:
            print(f"'{string}' not found in the text.")


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


# Perform the search
search_strings(sample_text)
```

```
'quick' found in the text.
'fox' found in the text.
'lazy' 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'

Solution:-- def search_word_in_text(text, word):

```
    indices = []
    index = text.find(word)
    while index != -1:
        indices.append(index)
        index = text.find(word, index + 1)
```

```
    return indices

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

locations = search_word_in_text(sample_text, searched_word)

if locations:
    print(f"The word '{searched_word}' is found at the following location(s): {locations}")
else:
    print(f"The word '{searched_word}' is not found in the text.")
```

```
The word 'fox' is found at the following 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'.

Solution:-- import re

```
def find_substrings_regex(text, substring):
    matches = re.finditer(substring, text)
    positions = [match.start() for match in matches]
    return positions

# Sample text and substring to find
sample_text = 'Python exercises, PHP exercises, C# exercises'
substring_to_find = 'exercises'

# Find occurrences of the substring in the text using regex
```

```
substrings_found = find_substrings_regex(sample_text, substring_to_find)
if substrings_found:
    print(f"The substring '{substring_to_find}' is found at positions: {substrings_found}")
else:
    print(f"The substring '{substring_to_find}' is not found in the text.")
```

```
The pattern 'exercises' is found at positions: [7, 22, 36]
```

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

Solution:-

```
def find_occurrences_positions(text, substring):
    start = 0
    occurrences = []
    while True:
        start = text.find(substring, start)
        if start == -1:
            break
        occurrences.append((substring, start))
        start += 1
    return occurrences

# Sample text and substring to find
sample_text = 'Python exercises, PHP exercises, C# exercises, Python is great for Pythonistas'
substring_to_find = 'Python'

# Find occurrences and positions of the substring in the text
substrings_occurrences_positions = find_occurrences_positions(sample_text, substring_to_find)
if substrings_occurrences_positions:
    print(f"The substring '{substring_to_find}' occurrences and positions:")
```

```
    for occurrence, position in substrings_occurrences_positions:
        print(f"Occurrence: {occurrence}, Position: {position}")
else:
    print(f"The substring '{substring_to_find}' is not found in the text.")
```

```
The substring 'Python' occurrences and positions:
Occurrence: Python, Position: 0
Occurrence: Python, Position: 47
Occurrence: Python, Position: 67
```

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

Solution:-- from datetime import datetime

```
def convert_date_format(date_str):
    # Convert string to datetime object using the input format yyyy-mm-dd
    date_obj = datetime.strptime(date_str, '%Y-%m-%d')

    # Convert the datetime object to the desired output format dd-mm-yyyy
    new_date_format = date_obj.strftime('%d-%m-%Y')
    return new_date_format

# Sample date in yyyy-mm-dd format
input_date = '2023-01-15'

# Convert the date format
output_date = convert_date_format(input_date)

print(f"Input date (yyyy-mm-dd): {input_date}")
print(f"Converted date (dd-mm-yyyy): {output_date}")
```

```
Input date (yyyy-mm-dd): 2023-01-15
```

```
Converted date (dd-mm-yyyy): 15-01-2023
```

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']

Solution:-- import re

```
def find_decimal_numbers(text):
    pattern = re.compile(r'\b\d+\.\d{1,2}\b')
    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 with precision of 1 or 2 in the text
result = find_decimal_numbers(sample_text)
print("Decimal numbers with precision of 1 or 2:", result)
```

```
Decimal numbers with precision of 1 or 2: ['01.12', '145.8', '3.01', '27.2
5', '0.25']
```

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

Solution:-- def separate_numbers_positions(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 their positions in the text
numbers_found, positions_found = separate_numbers_positions(sample_text)

print("Numbers found:", numbers_found)
print("Positions of numbers:", positions_found)
```

```
Numbers found: ['1', '2', '3', '4', '5', '6', '7', '8', '9']
Positions of numbers: [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

Solution:-- import re

```
def extract_maximum_number(text):
    numbers = re.findall(r'\d+', text)
    if numbers:
```

```
        max_number = max(map(int, 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 from the text
max_numeric_value = extract_maximum_number(sample_text)

if max_numeric_value is not None:
    print(f"The maximum numeric value in the text is: {max_numeric_value}")
else:
    print("No numeric values found in the text.")
```

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

Solution:-

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

# Sample Text
sample_text = "RegularExpressionIsAnImportantTopicInPython"

# Insert spaces between words starting with capital letters
result = insert_spaces(sample_text)
print("Modified text with spaces:", result)
```

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

Solution:-- import re

```
def find_uppercase_followed_by_lowercase(text):
    pattern = re.compile(r'[A-Z][a-z]+')
    sequences = pattern.findall(text)
    return sequences

# Sample Text
sample_text = "This Is a Sample text with Some Uppercase and Lowercase Letters"

# Find sequences of one uppercase letter followed by lowercase letters
result = find_uppercase_followed_by_lowercase(sample_text)
print("Sequences of one uppercase followed by lowercase letters:", result)
```

```
Sequences of one uppercase followed by lowercase letters: ['This', 'Is', 'Sam
ple', 'Some', 'Uppercase', 'Lowercase', 'Letters']
```

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

Solution:-- import re

```
def remove_continuous_duplicates(sentence):
    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 from the sentence

result = remove_continuous_duplicates(sample_text)

print("Sentence after removing continuous duplicates:", result)

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

Solution:-- import re

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

# Test strings
test_strings = [
    "Hello123",
    "world567",
    "12345",
    "SomeText@",
    "Alphanumeric1_",
    "Another Example!"
]

for string in test_strings:
    result = ends_with_alphanumeric(string)
    print(f"String: {string}, Ends with alphanumeric character: {result}")
```

```
String: Hello123, Ends with alphanumeric character: True
String: world567, Ends with alphanumeric character: True
String: 12345, Ends with alphanumeric character: True
String: SomeText@, Ends with alphanumeric character: False
String: Alphanumeric1_, Ends with alphanumeric character: True
String: Another Example!, Ends with alphanumeric character: False
```

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']

Solution:-- import re

```
def extract_hashtags(text):
    hashtags = re.findall(r'#\w+', 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 from the text
extracted_hashtags = extract_hashtags(sample_text)

print("Extracted Hashtags:")
print(extracted_hashtags)
```

```
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

Solution:-- import re

```
def remove_Uplus_symbols(text):
    cleaned_text = re.sub(r'<U\+[A-Fa-f0-9]+>', '', 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 from the text
cleaned_text = remove_Uplus_symbols(sample_text)
print("Text after removing <U+..> like symbols:")
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.

Solution:--

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.

Solution:-- import re

```
def remove_words_length_between_2_and_4(text):
    pattern = re.compile(r'\b\w{2,4}\b')
    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 from the text
cleaned_text = remove_words_length_between_2_and_4(sample_text)
print("Text after removing words of length between 2 and 4:")
print(cleaned_text)
```

```
ext after removing words of length between 2 and 4:
 following example creates  ArrayList  a capacity   elements. 4 elements
added   ArrayList   ArrayList  trimmed accordingly.
```