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_characters(input_text):
 # Define the characters to be replaced
    characters_to_replace = [' ', ',', '.']

 # Iterate through the characters and replace them with a colon
    for char in characters_to_replace:

        input_text = input_text.replace(char, ':')



return input_text

 # Sample text
sample_text = 'Python Exercises, PHP exercises.'

 # Replace characters and print the result
output_text = replace_characters(sample_text)

print ("Sample Text:", sample_text)
print ("Expected Output:", output_text)
```

Output:

Sample Text: Python Exercises, PHP exercises.

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

Ans:

```
import pandas as pd
import re
 # Dictionary
data = {'SUMMARY': ['hello, world!', 'XXXXX test', '123four, five:; six...']}

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

 # Function to remove unwanted characters
def clean_text(text):

 # Remove everything except words
cleaned_text = re.sub(r'[^a-zA-Z\s]', '', text)

return cleaned_text.strip()
```

```
 # Apply the cleaning function to the 'SUMMARY' column

df['SUMMARY'] = df['SUMMARY'].apply(clean_text)
 # Display the DataFrame

print(df)
```

Output:

```
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.

Ans:

```
import re
 def find_long_words(input_string):

    # Define a regex pattern for words with at least 4 characters
    pattern = re.compile(r'\b\w{4,}\b')

    # Use findall to extract words from the input string
words = pattern.findall(input_string)

return words

# Example usage
```

```
input_text = "This is a sample sentence with some words of varying lengths."

result = find_long_words(input_text)
 print ("Original text:", input_text)
print ("Words with at least 4 characters:", result)
```

Output:

Original text: This is a sample sentence with some words of varying lengths.

Words with at least 4 characters: ['This', 'sample', 'sentence', 'with', 'some', 'words', 'varying', 'lengths']

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_of_lengths(input_string):

    # Define a regex pattern for words with lengths 3, 4, and 5 characters
    pattern = re.compile(r'\b\w{3,5}\b')

    # Use findall to extract words from the input string
    words = pattern.findall(input_string)

    return words

# Example usage
input_text = "This is a sample sentence with some words of varying lengths."
result = find_words_of_lengths(input_text)
```

```
print ("Original text:", input_text)
print ("Three, four, and five character words:", result)
```

Output: Original text: This is a sample sentence with some words of varying lengths.

Three, four, and five character words: ['This', 'is', 'a', 'with', 'some', 'words', 'of']

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):
    # Define a regex pattern for parentheses and their contents
    pattern = re.compile(r'\([^)]*\)')

    # Use sub() to remove parentheses and their contents in each string
    result_list = [pattern.sub('', string).strip() for string in strings_list]

    return result_list
```

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

output_text = remove_parentheses(sample_text)

# Display the result
for original, cleaned in zip(sample_text, output_text):
    print(f"Original: {original}\nCleaned: {cleaned}\n")
```

Output: Original: example (.com)
Cleaned: example.com

Original: hr@fliprobo (.com)
Cleaned: hr@fliprobo.com

Original: github (.com)
Cleaned: github.com

Original: Hello (Data Science World)
Cleaned: Hello Data Science World

Original: Data (Scientist)
Cleaned: 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.

Ans:

```
import re
def remove_parentheses_from file(input_file, output_file):
    # Read text from the input file
    with open(input_file, 'r') as file:
        text = file.read()
# Define a regex pattern for parentheses and their contents
    pattern = re.compile(r'\([^)]*\)')

    # Use sub() to remove parentheses and their contents
    cleaned_text = pattern.sub('', text).strip()

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

# Sample usage
input_file_path = 'input_text.txt'
output_file_path = 'output_text.txt'
```

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

# Write sample text to the input file
with open(input_file_path, 'w') as file:
    for line in sample_text:
        file.write(line + '\n')

# Remove parentheses and write to the output file
remove_parentheses_from_file(input_file_path, output_file_path)

# Read and display the cleaned text from the output file
with open(output_file_path, 'r') as file:
    cleaned_text = file.read()
    print("Cleaned Text:", cleaned_text)
```

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_string_by_uppercase(input_string):
    # Use re.findall() to find all sequences of uppercase letters
    result = re.findall(r'[A-Z][a-z]*', input_string)
    return result

# Sample text
sample_text = "ImportanceOfRegularExpressionsInPython"

# Split the string by uppercase letters
output = split_string_by_uppercase(sample_text)

# Display the result
print ("Sample Text:", sample_text)
print("Expected Output:", output)
```

Output:

Sample Text: ImportanceOfRegularExpressionsInPython

Expected 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_before_numbers(input_string):
    # Use re.sub() to insert a space before words starting with numbers
    result = re.sub(r'(?<=[^\d\s])(?=\d)', ' ', input_string)
    return result
```

```
# Sample text
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Insert spaces before words starting with numbers
output = insert_spaces_before_numbers(sample_text)

# Display the result
print ("Sample Text:", sample_text)
print ("Expected Output:", output)
```

Output:

Sample Text: RegularExpression1IsAn2ImportantTopic3InPython

Expected 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_before_capital_and_numbers(input_string):
    # Use re.sub() to insert a space before words starting with capital letters or numbers
    result = re.sub(r'(?<=[a-z])(?=[A-Z0-9])|(?<=\D)(?=\d)', ' ', input_string)
    return result
 # Sample text
```

```
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
 # Insert spaces before words starting with capital letters or numbers
output = insert_spaces_before_capital_and_numbers(sample_text)
 # Display the result
print("Sample Text:", sample_text)
print("Expected Output:", output)
```

Output:

Sample Text: RegularExpression1IsAn2ImportantTopic3InPython

Expected 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


# Assuming you have a list of countries like this
countries = ["United States", "United Kingdom", "Canada", "Australia", "Germany", "France"]


# Create a dataframe
df = pd.DataFrame({"Country": countries})


# Extract the first 6 letters of each country
df['first_five_letters'] = df['Country'].str[:6]
```

```
# Display the dataframe
print(df)
```

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

Ans:

```
import re

def is_valid_string(s):
    # Define the regular expression pattern
    pattern = re.compile("^[a-zA-Z0-9_]+$")

    # Check if the string matches the pattern
    if pattern.match(s):
        return True
    else:
        return False

# Example usage:
input_string = "Hello_World123"
if is_valid_string(input_string):
    print (f"The string '{input_string}' is valid.")
else:
    print (f"The string '{input_string}' is not valid.")
```

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

Ans:

```
def starts_with_number(input_string, target_number):
    # Check if the string starts with the specified number
    return input_string.startswith(str(target_number))
 # Example usage:
input_string = "123HelloWorld"
target_number = 123

if starts_with_number(input_string, target_number):
    print (f"The string '{input_string}' starts with the number {target_number}.")
else:
    print (f"The string '{input_string}' does not start with the number {target_number}.")
```

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

Ans:

```
import ipaddress

def remove_leading_zeros(ip_address):
    # Validate the input IP address
    try:
        ip = ipaddress.ip_address(ip_address)
    except ValueError:
        return "Invalid IP address"

    # Convert the IP address to string, removing leading zeros
    cleaned_ip = ".".join(str(int(octet)) for octet in ip.packed)

    return cleaned_ip
```

```
# Example usage:
ip_with_zeros = "192.012.003.004"
cleaned_ip = remove_leading_zeros(ip_with_zeros)

print(f"Original IP: {ip_with_zeros}")
print (f"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

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

# Example usage:
file_path = 'your_text_file.txt'

with open(file_path, 'r') as file:
    content = file.read()
```

```
    matches = date_pattern.findall(content)

if matches:
    print("Found date strings:")
    for match in matches:
        print(match)
else. print ("No date strings found.")
```

Question 15- Write a Python program to search some literals strings in a string.

Sample text : 'The quick brown fox jumps over the lazy dog.'

Searched words : 'fox', 'dog', 'horse'

Ans:

```
def search_literals(text, search_words):
    found_words = [word for word in search_words if word in text]
    return found_words

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

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

# Search for the words in the sample text
result = search_literals(sample_text, searched_words)

# Display the result
print ("Found words:", result)
```

Output: Found words: ['fox', 'dog']

Question 16- Write a Python program to search a literals string in a string and also find the location within the original string where the pattern occurs

Sample text : 'The quick brown fox jumps over the lazy dog.'

Searched words : 'fox'

Ans:

```
def search_and_locate(text, search_word):
    occurrences = []
    start_index = 0

    while start_index < len(text):
        index = text.find(search_word, start_index)
        if index == -1:
            break
        occurrences.append(index)
        start_index = index + 1

    return occurrences

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

# Word to search for
searched_word = 'fox'

# Search for the word in the sample text and find its locations
result = search_and_locate(sample_text, searched_word)
```

```
# Display the result
if result:
    print (f"The word '{searched_word}' found at positions: {result}")
else:
    print (f"The word '{searched_word}' not found in the text.")
```

Output:

The word 'fox' found at positions: [16]

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):
    occurrences = []
    start_index = 0

    while start_index < len(text):
        index = text.find(pattern, start_index)
        if index == -1:
            break
        occurrences.append(index)
        start_index = index + 1

    return occurrences
```

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

# Substring to search for
pattern = 'exercises'

# Find occurrences of the substring in the sample text
result = find_substrings(sample_text, pattern)

# Display the result
if result:
    print(f"The substring '{pattern}' found at positions: {result}")
else:
    print(f"The substring '{pattern}' not found in the text.")
```

Output:

The substring 'exercises' found at positions: [7, 24, 39]

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

Ans:

```
def find_occurrences_and_positions(text, pattern):
    occurrences = []
    start_index = 0

    while start_index < len(text):
        index = text.find(pattern, start_index)
        if index == -1:
            break
```

```
        occurrences.append((index, index + len(pattern) - 1))
        start_index = index + 1

    return occurrences

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

# Substring to search for
pattern = 'fox'

# Find occurrences and positions of the substring in the sample text
result = find_occurrences_and_positions(sample_text, pattern)

# Display the result
if result:
    print (f"The substring '{pattern}' found at positions: {result}")
else:
    print (f"The substring '{pattern}' not found in the text.")
```

Output:

The substring 'fox' found at positions: [(16, 18)]

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

from datetime import datetime

Ans:

```
def convert_date_format(input_date):
    # Parse the input date in yyyy-mm-dd format
    input_datetime = datetime.strptime(input_date, '%Y-%m-%d')
```

```
    # Convert the date to dd-mm-yyyy format
    output_date = input_datetime.strftime('%d-%m-%Y')

    return output_date

# Example: Convert '2024-01-28' to '28-01-2024'
input_date = '2024-01-28'
output_date = convert_date_format(input_date)

print (f"Original date: {input_date}")
print (f"Converted date: {output_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):
    # Define the regular expression pattern
    pattern = re.compile(r'\b\d+\.\d{1,2}\b')

    # Use findall to extract all matching decimal numbers
    result = pattern.findall(text)

    return result

# Sample Text
```

```
sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

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

# Print the result
print ("Sample Text:", sample_text)
print ("Decimal Numbers:", decimal_numbers)
```

Output:

Sample Text: 01.12 0132.123 2.31875 145.8 3.01 27.25 0.25

Decimal Numbers: ['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 extract_numbers_with_position(input_string):
    result = []

    # Iterate through each character and track the position
    for position, char in enumerate(input_string):
        # Check if the character is a digit
        if char.isdigit():
            # Find the extent of the number
            start = position
            end = position + 1
            while end < len(input_string) and input_string[end].isdigit():
```

```
            end += 1

        # Extract the number and its position
        number = input_string[start:end]
        result.append((number, start))

    return result

# Sample Text
sample_text = "abc 123 def 456 xyz 789"

# Extract numbers with their positions
numbers_with_positions = extract_numbers_with_position(sample_text)

# Print the result
print ("Sample Text:", sample_text)
print ("Numbers with Positions:", numbers_with_positions)
```

Output:

Sample Text: abc 123 def 456 xyz 789

Numbers with Positions: [('123', 4), ('456', 12), ('789', 20)]

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_maximum_numeric_value(input_string):
    # Find all numeric values in the string
```

```
    numeric_values = re.findall(r'\b\d+\b', input_string)

    if not numeric_values:
        return None

    # Convert the numeric values to integers and find the maximum
    max_numeric_value = max(map(int, numeric_values))

    return max_numeric_value

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

# Extract maximum numeric value
max_value = extract_maximum_numeric_value(sample_text)

# Print the result
print ("Sample Text:", sample_text)
print ("Maximum Numeric Value:", max_value)
```

Ans:

Sample Text: My marks in each semester are: 947, 896, 926, 524, 734, 950, 642

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(input_string):
    # Use regular expression to insert space before capital letters
    result_string = re.sub(r'([a-z])([A-Z])', r'\1 \2', input_string)

    # Capitalize the first letter of the result string
    result_string = result_string.capitalize()

    return result_string

# Sample Text
sample_text = "RegularExpressionIsAnImportantTopicInPython"

# Insert spaces
output_text = insert_spaces(sample_text)

# Print the result
print ("Sample Text:", sample_text)
print ("Output Text:", output_text)
```

Output:

Sample Text: RegularExpressionIsAnImportantTopicInPython

Output 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

text = "Hello World and Regular Expressions in Python"

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

matches = pattern.findall(text)

print(matches)
```

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(sentence):
    # Use re.sub with a pattern to remove continuous duplicate words
    cleaned_sentence = re.sub(r'\b(\w+)( \1\b)+', r'\1', sentence)
    return cleaned_sentence

# Sample Text
sample_text = "Hello hello world world"

# Remove continuous duplicate words
```

```
result = remove_continuous_duplicates(sample_text)

# Display the result
print ("Original Sentence:", sample_text)
print ("Processed Sentence:", result)
```

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

Ans:

```
import re

def is_string_ending_with_alphanumeric(input_string):
    # Define the regular expression pattern
    pattern = re.compile(r'^.*[a-zA-Z0-9]$')

    # Use re.match to check if the input string matches the pattern
    match = pattern.match(input_string)

    # Return True if there is a match, otherwise False
    return bool(match)

# Test the function
test_string1 = "Hello123"
test_string2 = "Test@123!"

print(f'Is "{test_string1}" ending with alphanumeric?
{is_string_ending_with_alphanumeric(test_string1)}')
print(f'Is "{test_string2}" ending with alphanumeric?
{is_string_ending_with_alphanumeric(test_string2)}')
```

Output:

Is "Hello123" ending with alphanumeric? True

Is "Test@123!" ending with alphanumeric? 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']

Ans:

```
import re

def extract_hashtags(input_text):
    # Define the regular expression pattern to match hashtags
    pattern = re.compile(r'#\w+')

    # Use re.findall to find all occurrences of the pattern in the text
    hashtags = re.findall(pattern, input_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 sample text
result = extract_hashtags(sample_text)
```

Print the result

print ("Extracted Hashtags:", result)

Ans:

```
import re

def remove_unicode_symbols(input_text):
    # Define the regular expression pattern to match <U+..> like symbols
    pattern = re.compile(r'<U\+[0-9A-Fa-f]+>')

    # Use re.sub to replace all occurrences of the pattern with an empty string
    result_text = re.sub(pattern, '', input_text)

    return result_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 sample text
result = remove_unicode_symbols(sample_text)

# Print the result
print ("Result Text:", result)
```

Output:

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

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(input_text):
    # Define the regular expression pattern to match <U+..> like symbols
    pattern = re.compile(r'<U\+[0-9A-Fa-f]+>')

    # Use re.sub to replace all occurrences of the pattern with an empty string
    result_text = re.sub(pattern, '', input_text)

    return result_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 sample text
```

```
result = remove_unicode_symbols(sample_text)

# Print the result
print ("Result Text:", result)
```

Output:

Result 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_path):
    # Read text from the file
    with open (file_path, 'r') as file:
        text = file.read()

    # Define the regular expression pattern to match dates in the format DD-MM-YYYY
    pattern = re.compile(r'\b\d{2}-\d{2}-\d{4}\b')

    # Use re.findall to extract dates from the text
    dates = re.findall(pattern, text)

    return dates
```

```
# Store the sample text in a file
file_path = 'sample_text.txt'
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 from the file
extracted_dates = extract_dates_from_text(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.

Ans:
```
import re

def remove_words_of_length_between_2_and_4(input_text):
    # Define the regular expression pattern to match words of length 2 to 4
    pattern = re.compile(r'\b\w{2,4}\b')

    # Use re.sub to replace matched words with an empty string
    result_text = re.sub(pattern, '', input_text)

    return result_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
output_text = remove_words_of_length_between_2_and_4(sample_text)

# Print the result
print ("Original Text:", sample_text)
print ("Output Text:", output_text)
```