**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 1 -   
import re

# Sample text

text = 'Python Exercises, PHP exercises.'

# Replace space, comma, or dot with a colon

result = re.sub(r'[ ,.]', ':', text)

# Output the result

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 2  
import pandas as pd

import re

# Given dictionary

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

# Create DataFrame

df = pd.DataFrame(data)

# Function to clean the text

def clean\_text(text):

# Remove everything except words

return ' '.join(re.findall(r'\b\w+\b', text))

# Apply the function to the SUMMARY column

df['SUMMARY'] = df['SUMMARY'].apply(clean\_text)

# Output the result

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 3  
  
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 words that match the pattern

long\_words = pattern.findall(text)

return long\_words

# Example usage

text = "This is a test string with several words of various lengths."

result = find\_long\_words(text)

print(result)  
  
Output  
  
['This', 'test', 'string', 'with', 'several', 'words', 'various', '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 4-  
import re

def find\_specific\_length\_words(text):

# Compile a regular expression pattern to match words with 3, 4, or 5 characters

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

# Find all words that match the pattern

specific\_length\_words = pattern.findall(text)

return specific\_length\_words

# Example usage

text = "This is a test string with several words of various lengths."

result = find\_specific\_length\_words(text)

print(result)  
  
Output-  
['This', 'test', 'with', 'words']

**Question 5-** Create a function in Python to remove the parenthesis in a list of strings. The use of the re.compile() method is mandatory.

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

**Expected Output:**

example.com

hr@fliprobo.com

github.com

Hello Data Science World

Data Scientist

Ans 5  
import re

def remove\_parentheses(text\_list):

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

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

# Remove the matched patterns from each string in the list

cleaned\_list = [pattern.sub('', text).strip() for text in text\_list]

return cleaned\_list

# Sample text

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

# Remove parentheses

result = remove\_parentheses(text\_list)

print(result)

**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 6 -  
import re

def remove\_parentheses\_from\_file(file\_path):

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

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

# Read the text from the file

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

lines = file.readlines()

# Remove the matched patterns from each line

cleaned\_lines = [pattern.sub('', line).strip() for line in lines]

# Write the cleaned text back to the file or print the result

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

for line in cleaned\_lines:

file.write(line + '\n')

# Optional: print the cleaned lines

print(cleaned\_lines)

# Sample file path

file\_path = 'sample.txt'

# Remove parentheses from the file

remove\_parentheses\_from\_file(file\_path)

**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 7  
import re

def split\_by\_uppercase(text):

# Compile a regular expression pattern to split by uppercase letters

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

# Split the text using the pattern

result = pattern.split(text)

return result

# Sample text

text = "ImportanceOfRegularExpressionsInPython"

# Split the text by uppercase letters

result = split\_by\_uppercase(text)

print(result)

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

Sample Text: “RegularExpression1IsAn2ImportantTopic3InPython"

Expected Output: RegularExpression 1IsAn 2ImportantTopic 3InPython  
  
Ans 8-  
import re

def insert\_spaces(text):

# Compile a regular expression pattern to match numbers followed by words

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

# Insert a space before the matched pattern

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

return result

# Sample text

text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Insert spaces between words starting with numbers

result = insert\_spaces(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  
  
ans 9-

import re

def insert\_spaces(text):

# Compile a regular expression pattern to match uppercase letters or numbers

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

# Insert a space before the matched pattern

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

return result.strip()

# Sample text

text = "RegularExpression1IsAn2ImportantTopic3InPython"

# Insert spaces between words starting with capital letters or numbers

result = insert\_spaces(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>

Ans 10-  
import csv

# Define the file path

file\_path = 'world\_happiness\_report.csv'

# Initialize an empty list to store the data

happiness\_data = []

# Read data from CSV file

with open(file\_path, mode='r', encoding='utf-8') as file:

csv\_reader = csv.DictReader(file)

for row in csv\_reader:

# Convert numerical values to appropriate types

for key in ['Happiness Rank', 'Happiness Score', 'Standard Error', 'Economy (GDP per Capita)',

'Family', 'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',

'Generosity', 'Dystopia Residual']:

row[key] = float(row[key])

# Append the processed row to the list

happiness\_data.append(row)

# Display the first few entries to verify

for i in range(5):

print(happiness\_data[i])

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

import re

def match\_string(input\_string):

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

if re.match(pattern, input\_string):

return True

else:

return False

# Test cases

strings\_to\_test = [

"Hello\_World123",

"Python\_3",

"abc123",

"12345",

"no spaces",

"with space",

"special@char",

]

for string in strings\_to\_test:

if match\_string(string):

print(f"{string}: Matched")

else:

print(f"{string}: Not matched")

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

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

# Convert number to string to compare easily

number\_str = str(number)

# Check if the string starts with the number

if string.startswith(number\_str):

return True

else:

return False

# Example usage:

input\_string = "123abc"

specific\_number = 123

if starts\_with\_number(input\_string, specific\_number):

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

else:

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

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

def remove\_leading\_zeros(ip\_address):

# Split the IP address into its octets

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

# Remove leading zeros from each octet

cleaned\_octets = [str(int(octet)) for octet in octets]

# Join the cleaned octets back into an IP address

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

return cleaned\_ip

# Example usage:

ip\_address\_with\_zeros = "192.168.001.001"

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

print(f"Original IP Address: {ip\_address\_with\_zeros}")

print(f"Cleaned IP Address: {cleaned\_ip\_address}")

**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 14 -   
import re

# Function to extract date string from text file

def extract\_date\_from\_file(file\_path):

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

text = file.read()

# Define regular expression pattern to match month name followed by day number and year

pattern = 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'

# Search for the pattern in the text

match = re.search(pattern, text)

if match:

return match.group(0)

else:

return "Date string not found in the text."

# Example usage:

file\_path = 'sample\_text.txt' # Replace with your file path

date\_string = extract\_date\_from\_file(file\_path)

print("Extracted Date String:", date\_string)

**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 15 -  
def search\_literals(text, searched\_words):

found\_words = []

for word in searched\_words:

if word in text:

found\_words.append(word)

return found\_words

# Sample text

sample\_text = 'The quick brown fox jumps over the lazy dog.'

# Words to search

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

# Search for words in the sample text

found\_words = search\_literals(sample\_text, searched\_words)

# Print the found words

print("Found words:", found\_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'  
  
Ans 16 -   
def search\_and\_find(text, pattern):

index = text.find(pattern)

if index != -1:

return f"'{pattern}' found at index {index}"

else:

return f"'{pattern}' not found in the text"

# Sample text

sample\_text = 'The quick brown fox jumps over the lazy dog.'

# Pattern to search

pattern = 'fox'

# Search and find the pattern in the sample text

result = search\_and\_find(sample\_text, pattern)

# Print the result

print(result)

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

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

**Pattern :** 'exercises'.  
  
Ans 17 -   
  
def find\_substrings(text, pattern):

start = 0

indices = []

while True:

start = text.find(pattern, start)

if start == -1:

break

indices.append(start)

start += 1

return indices

# Sample text

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

# Pattern to search

pattern = 'exercises'

# Find all occurrences of the pattern in the sample text

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

# Print the result

print(f"Occurrences of '{pattern}' at positions:", occurrences)

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

Ans 18 -   
def find\_occurrences\_positions(text, pattern):

start = 0

occurrences = []

while True:

start = text.find(pattern, start)

if start == -1:

break

occurrences.append(start)

start += 1

return occurrences, len(occurrences)

# Sample text

sample\_text = 'Python exercises, PHP exercises, C# exercises, Python is fun.'

# Pattern to search

pattern = 'Python'

# Find occurrences and positions of the pattern in the sample text

positions, count = find\_occurrences\_positions(sample\_text, pattern)

# Print the result

print(f"Pattern '{pattern}' found {count} times at positions:", positions)

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

Ans 19-  
def convert\_date\_format(date\_str):

# Split the date into year, month, and day components

parts = date\_str.split('-')

# Rearrange the parts to form the new date format

new\_date\_str = f"{parts[2]}-{parts[1]}-{parts[0]}"

return new\_date\_str

# Example usage:

date\_in\_yyyy\_mm\_dd = '2023-07-04'

converted\_date = convert\_date\_format(date\_in\_yyyy\_mm\_dd)

print(f"Original date: {date\_in\_yyyy\_mm\_dd}")

print(f"Converted date: {converted\_date}")

**Question 20-** Create a function in python to find all decimal numbers with a precision of 1 or 2 in a string. The use of the re.compile() method is mandatory.

**Sample Text:** "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

**Expected Output:** ['01.12', '145.8', '3.01', '27.25', '0.25']

Ans 20-  
  
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 sample text

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

# Print the result

print("Decimal numbers with precision of 1 or 2:", decimal\_numbers)

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

def print\_numbers\_and\_positions(text):

numbers = []

for index, char in enumerate(text):

if char.isdigit():

numbers.append((char, index))

return numbers

# Example usage:

sample\_text = "Hello 123 World 456"

numbers\_and\_positions = print\_numbers\_and\_positions(sample\_text)

# Print the numbers and their positions

for number, position in numbers\_and\_positions:

print(f"Number '{number}' found at position {position}")

**Question 22-** Write a regular expression in python program to extract maximum/largest numeric value from a string.

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

**Expected Output:** 950

Ans 22-  
import re

def extract\_maximum\_numeric\_value(text):

# Regular expression to match numbers

pattern = r'\b\d+\b'

# Find all numbers in the text

numbers = re.findall(pattern, text)

# Convert numbers from strings to integers

numbers = [int(num) for num in numbers]

# Find the maximum number

max\_number = max(numbers)

return max\_number

# Sample text

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

# Extract maximum numeric value from the sample text

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

# Print the result

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

**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 23-  
def insert\_spaces(text):

words = []

start = 0

for end in range(1, len(text)):

if text[end].isupper():

words.append(text[start:end])

start = end

# Append the last word

words.append(text[start:])

# Join words with spaces and return

return ' '.join(words)

# Example usage:

sample\_text = "RegularExpressionIsAnImportantTopicInPython"

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

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

print("Output Text:", output\_text)

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

Ans 24-  
import re

def find\_sequences(text):

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

sequences = re.findall(pattern, text)

return sequences

# Example usage:

sample\_text = "Hello World. This is a Test String With Sequences like This One and Another One."

found\_sequences = find\_sequences(sample\_text)

# Print the found sequences

print("Found sequences:", found\_sequences)

**Question 25-** Write a Python program to remove continuous duplicate words from Sentence using Regular Expression.

**Sample Text:** "Hello hello world world"

**Expected Output:** Hello hello world  
  
Ans 25-  
import re

def remove\_continuous\_duplicates(sentence):

# Use regex to remove continuous duplicate words

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

return cleaned\_sentence

# Example usage:

sample\_text = "Hello hello world world"

cleaned\_text = remove\_continuous\_duplicates(sample\_text)

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

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

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

def ends\_with\_alphanumeric(input\_string):

pattern = r'^.\*[a-zA-Z0-9]$' # Pattern to match strings ending with alphanumeric character

if re.match(pattern, input\_string):

return True

else:

return False

# Example usage:

test\_string1 = "Hello123"

test\_string2 = "Python!"

test\_string3 = "Testing123?"

# Check if strings end with alphanumeric character

print(f"{test\_string1}: {ends\_with\_alphanumeric(test\_string1)}")

print(f"{test\_string2}: {ends\_with\_alphanumeric(test\_string2)}")

print(f"{test\_string3}: {ends\_with\_alphanumeric(test\_string3)}")

**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 27- import re

def extract\_hashtags(text):

pattern = r'#\w+' # Pattern to match hashtags starting with #

hashtags = re.findall(pattern, text)

return hashtags

# Example usage:

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

hashtags = extract\_hashtags(sample\_text)

# Print the result

print("Extracted Hashtags:", hashtags)

**Question 28-** Write a python program using RegEx to remove <U+..> like symbols

Check the below sample text, there are strange symbols something of the sort <U+..> all over the place. You need to come up with a general Regex expression that will cover all such symbols.

**Sample Text:** "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are protesting #demonetization are all different party leaders"

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

Ans 28-

import re

def remove\_u\_symbols(text):

pattern = r'<U\+[0-9A-Fa-f]+>' # Pattern to match <U+..> symbols

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

return cleaned\_text

# Example usage:

sample\_text = "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who are protesting #demonetization are all different party leaders"

# Remove <U+..> symbols from the sample text

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

# Print the result

print("Original Text:")

print(sample\_text)

print("\nCleaned Text:")

print(cleaned\_text)

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

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

**Note-** Store this sample text in the file and then extract dates.  
ans 29-  
import re

def extract\_dates\_from\_file(file\_path):

# Read text from the file

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

text = file.read()

# Regular expression pattern to match dates in dd-mm-yyyy format

pattern = r'\b\d{2}-\d{2}-\d{4}\b'

# Find all dates matching the pattern

dates = re.findall(pattern, text)

return dates

# File path where the sample text is stored

file\_path = 'sample\_text.txt'

# Extract dates from the file

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.  
Ans 30-

import re

def remove\_words\_of\_length\_between\_2\_and\_4(text):

pattern = re.compile(r'\b\w{2,4}\b') # Compile pattern to match words of length 2 to 4

cleaned\_text = pattern.sub('', text) # Remove matched words from the text

cleaned\_text = re.sub(r'\s+', ' ', cleaned\_text).strip() # Remove extra spaces and strip leading/trailing spaces

return cleaned\_text

# Example usage:

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 sample text

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

# Print the result

print("Original Text:")

print(sample\_text)

print("\nCleaned Text:")

print(cleaned\_text)