Q-1
```
import re

text='Python Exercises, PHP exercises.'
result=re.sub(r'[ ,.]', ':',text)
print(result)
```

Output- Python:Exercises::PHP:exercises:

Q-2
```
import pandas as pd

import re

data = {'SUMMARY': ['hello, world!', 'Second test', '123four, five:; six...']
}
df=pd.DataFrame(data)
df['SUMMARY'] = df['SUMMARY'].apply(lambda x: re.sub(r'[^a-zA-Z\s]', '', x))
print(df)
```

Output-
```
         SUMMARY
0    hello world
1    Second test
2  four five six
```

Q-3
```
import re

def find_words(text):
    pattern = re.compile(r'\b\w{4,}\b')
    return pattern.findall(text)
text="This is only a test string with several words of varying length"
print(find_words(text))
```

Output- ['This', 'only', 'test', 'string', 'with', 'several', 'words', 'varying', 'length']

Q-4
```
import re

def remove_parentheses(lst):
    pattern = re.compile(r'\s*\(.*?\)\s*')
    return [pattern.sub('', item) for item in lst]

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

Output- ['example', 'hr@fliprobo', 'github', 'Hello', 'Data']

Q-5

```
import re

with open('sample_text.txt', 'w') as f:
    f.write(str(["example (.com)", "hr@fliprobo (.com)", "github (.com)",
"Hello (Data Science World)", "Data (Scientist)"]))

with open('sample_text.txt', 'r') as f:
    content = f.read()

pattern = re.compile(r'\s*\(.*?\)\s*')
result = pattern.sub('', content)
print(result)
```

Output- ['example', 'hr@fliprobo', 'github', 'Hello', 'Data']

Q-6

```
import re

def split_uppercase(text):
    pattern = re.compile(r'(?=[A-Z])')
    return pattern.split(text)

sample_text = "ImportanceOfRegularExpressionsInPython"
print(split_uppercase(sample_text))
```

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

Q-7

```
import re

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

text = "This is a test string with several words of varying length"
print(find_specific_length_words(text))
```

Output- ['This', 'test', 'with', 'words']

Q-8

```
import re

def insert_spaces_numbers(text):
    pattern = re.compile(r'(\d)')
    return pattern.sub(r' \1', text)

sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
print(insert_spaces_numbers(sample_text))
```

Output- RegularExpression 1IsAn 2ImportantTopic 3InPython

Q-9
```
import re

def insert_spaces_numbers_caps(text):
    pattern = re.compile(r'([A-Z0-9])')
    return pattern.sub(r' \1', text).strip()

sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
print(insert_spaces_numbers_caps(sample_text))
```

Output- Regular Expression 1 Is An 2 Important Topic 3 In Python

Q-10
```
import pandas as pd

url =
'https://raw.githubusercontent.com/dsrscientist/DSData/master/happiness_score
_dataset.csv'
df = pd.read_csv(url)

df['first_six_letters'] = df['Country'].str[:6]
print(df[['Country', 'first_six_letters']])
```

Output-
```
       Country first_six_letters
0    Switzerland            Switze
1        Iceland            Icelan
2        Denmark            Denmar
3         Norway            Norway
4         Canada            Canada
..           ...               ...
153       Rwanda            Rwanda
154        Benin             Benin
155        Syria             Syria
156      Burundi            Burund
157         Togo              Togo

[158 rows x 2 columns]
```

Q-11
```
import re

def match_string(text):
    pattern = re.compile(r'^\w+$')
    return bool(pattern.match(text))

text = "Valid_string_123"
```

```
print(match_string(text))  # True

text = "Invalid string!"
print(match_string(text))  # False
```

Output- True
False

Q-12
```
import re

def starts_with_number(text, number):
    pattern = re.compile(r'^' + str(number))
    return bool(pattern.match(text))

text = "1234 is the start"
print(starts_with_number(text, 1234))  # True

text = "5678 comes next"
print(starts_with_number(text, 1234))  # False
```

Output- True
False

Q-13
```
import re

def remove_leading_zeros(ip):
    pattern = re.compile(r'\b0+(\d)')
    return pattern.sub(r'\1', ip)

ip_address = "192.168.001.001"
print(remove_leading_zeros(ip_address))
```

Output- 192.168.1.1

Q-14
```
import re

with open('sample_text.txt', 'w') as f:
    f.write('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.')

with open('sample_text.txt', 'r') as f:
    content = f.read()
```

```
pattern = re.compile(r'\b[A-Z][a-z]+\s\d{1,2}\w{2}\s\d{4}\b')
result = pattern.search(content)
print(result.group())
```

Output- August 15th 1947

Q-15

```
import re

def search_literals(text, words):
    pattern = re.compile('|'.join(words))
    return pattern.findall(text)

text = 'The quick brown fox jumps over the lazy dog.'
searched_words = ['fox', 'dog', 'horse']
print(search_literals(text, searched_words))
```

Output- ['fox', 'dog']

Q-16

```
import re

def search_literals_with_location(text, word):
    pattern = re.compile(word)
    match = pattern.search(text)
    if match:
        return match.group(), match.start(), match.end()
    return None

text = 'The quick brown fox jumps over the lazy dog.'
searched_word = 'fox'
print(search_literals_with_location(text, searched_word))
```

Output- ('fox', 16, 19)

Q-17

```
import re

def find_substrings(text, pattern):
    pattern = re.compile(pattern)
    return pattern.findall(text)

text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'
print(find_substrings(text, pattern))
```

Output- ['exercises', 'exercises', 'exercises']

Q-18

```
import re

def find_substring_positions(text, pattern):
    matches = [(match.start(), match.end()) for match in re.finditer(pattern,
text)]
    return matches

text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'
print(find_substring_positions(text, pattern))
```

Output- [(7, 16), (22, 31), (36, 45)]

Q-19

```
import re

def convert_date_format(date_str):
    pattern = re.compile(r'(\d{4})-(\d{2})-(\d{2})')
    return pattern.sub(r'\3-\2-\1', date_str)

date = "2023-08-24"
print(convert_date_format(date))
```

Output- 24-08-2023

Q-20

```
import re

def find_decimal_numbers(text):
    pattern = re.compile(r'\b\d+\.\d{1,2}\b')
    return pattern.findall(text)

sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"
print(find_decimal_numbers(sample_text))
```

Output- ['01.12', '145.8', '3.01', '27.25', '0.25']

Q-21

```
import re

def separate_numbers_positions(text):
    pattern = re.compile(r'\d+')
    return [(match.group(), match.start(), match.end()) for match in
pattern.finditer(text)]
```

```
text = "The order was placed for 100 items at 5:30 PM on 2023-08-24."
print(separate_numbers_positions(text))
```

Output- [('100', 25, 28), ('5', 38, 39), ('30', 40, 42), ('2023', 49, 53), ('08', 54, 56), ('24', 57, 59)]

Q-22

```
import re

def extract_max_numeric_value(text):
    pattern = re.compile(r'\d+')
    numbers = list(map(int, pattern.findall(text)))
    return max(numbers)

sample_text = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'
print(extract_max_numeric_value(sample_text))
```

Output- 950

Q-23

```
import re

def insert_spaces_caps(text):
    pattern = re.compile(r'(?=[A-Z])')
    return pattern.sub(' ', text).strip()

sample_text = "RegularExpressionIsAnImportantTopicInPython"
print(insert_spaces_caps(sample_text))
```

Output- Regular Expression Is An Important Topic In Python

Q-24

```
import re

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

sample_text = "Here are some Words like Hello, World, and Python."
print(find_uppercase_followed_by_lowercase(sample_text))
```

Ouutp- ['Here', 'Words', 'Hello', 'World', 'Python']

Q-25

```
import re
```

```
def remove_duplicate_words(text):
    pattern = re.compile(r'\b(\w+)\b(?:\s+\1\b)+', re.IGNORECASE)
    return pattern.sub(r'\1', text)

sample_text = "Hello hello world world"
print(remove_duplicate_words(sample_text))
```

Output- Hello world

Q-26
```
import re

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

sample_text = "HelloWorld123"
print(ends_with_alphanumeric(sample_text))  # True

sample_text = "HelloWorld!"
print(ends_with_alphanumeric(sample_text))  # False
```

Output- True
False

Q-27
```
import re

def extract_hashtags(text):
    pattern = re.compile(r'#\w+')
    return pattern.findall(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"""
print(extract_hashtags(sample_text))
```

Output- ['#Doltiwal', '#xyzabc', '#Demonetization']

Q-28
```
import re

def remove_unicode_symbols(text):
    pattern = re.compile(r'<U\+\w+>')
    return pattern.sub('', 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"
print(remove_unicode_symbols(sample_text))
```

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

Q-29

```
import re

# Write the sample text to a file
with open('sample_text.txt', 'w') as f:
    f.write("Ron was born on 12-09-1992 and he was admitted to school on 15-12-1999.")

# Read the text from the file and extract dates
with open('sample_text.txt', 'r') as f:
    content = f.read()

pattern = re.compile(r'\d{2}-\d{2}-\d{4}')
dates = pattern.findall(content)
print(dates)
```

Output- ['12-09-1992', '15-12-1999']

Q-30

```
import re

def remove_short_words(text):
    pattern = re.compile(r'\b\w{2,4}\b')
    return pattern.sub('', 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."
result = remove_short_words(sample_text)
result = re.sub(r'\s{2,}', ' ', result).strip()
print(result)
```

Output- following example creates ArrayList a capacity elements. 4 elements added ArrayList ArrayList trimmed accordingly.