Regular Expressions

Question 1- Write a Python program to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

Answer: import re

```
def validate_string(input_string):
    pattern = r'^[a-zA-Z0-9]+$'
    if re.match(pattern, input_string):
        return True
    else:
        return False

# Example usage
string1 = "Hello123"
string2 = "Hello@123"

print(validate_string(string1))  # Output: True
print(validate_string(string2))  # Output: False
```

Question 2- Create a function in python that matches a string that has an a followed by zero or more b's

Answer: import re

```
def match_string(input_string):
    pattern = r'^ab*$'
    if re.match(pattern, input_string):
        return True
    else:
```

```
    return False


# Example usage
string1 = "abbbb"
string2 = "accc"
string3 = "abb"

print(match_string(string1))  # Output: True
print(match_string(string2))  # Output: False
print(match_string(string3))  # Output: True
```

Question 3- Create a function in python that matches a string that has an a followed by one or more b's

Answer: import re

```
def match_string(input_string):
    pattern = r'^ab+$'
    if re.match(pattern, input_string):
        return True
    else:
        return False

# Example usage
string1 = "abbbb"
string2 = "accc"
string3 = "abb"

print(match_string(string1))  # Output: True
print(match_string(string2))  # Output: False
print(match_string(string3))  # Output: False
```

Question 4- Create a function in Python and use RegEx that matches a string that has an a followed by zero or one 'b'.

Answer: import re

```
def match_string(input_string):
    pattern = r'^ab?$'
    if re.match(pattern, input_string):
        return True
    else:
        return False

# Example usage
string1 = "ab"
string2 = "a"
string3 = "ac"
string4 = "abb"

print(match_string(string1))  # Output: True
print(match_string(string2))  # Output: True
print(match_string(string3))  # Output: False
print(match_string(string4))  # Output: False
```

Question 5- Write a Python program that matches a string that has an a followed by three 'b'.

Answer: import re

```
def match_string(input_string):
    pattern = r'^abb$'
    if re.match(pattern, input_string):
        return True
    else:
```

```
    return False


# Example usage
string1 = "abb"
string2 = "abbb"
string3 = "aabbb"
string4 = "abc"

print(match_string(string1))  # Output: True
print(match_string(string2))  # Output: False
print(match_string(string3))  # Output: False
print(match_string(string4))  # Output: False
```

Question 6- Write a regular expression in Python to split a string into uppercase letters.

Sample text: “ImportanceOfRegularExpressionsInPython”

Output: [‘Importance’, ‘Of’, ‘Regular’, ‘Expression’, ‘In’, ‘Python’]

Answer: import re

```
def split_string_by_uppercase(input_string):
   pattern = r'[A-Z][a-z]*'
   uppercase_words = re.findall(pattern, input_string)
   return uppercase_words

# Example usage
input_text = "ImportanceOfRegularExpressionsInPython"
result = split_string_by_uppercase(input_text)
print(result)  # Output: ['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

Question 7- Write a Python program that matches a string that has an a followed by two to three 'b'.

Answer: import re

```
def match_string(input_string):
    pattern = r'^ab{2,3}$'
    if re.match(pattern, input_string):
        return True
    else:
        return False

# Example usage
string1 = "abb"
string2 = "abbb"
string3 = "abbbb"
string4 = "aabbb"
string5 = "abc"

print(match_string(string1))  # Output: True
print(match_string(string2))  # Output: True
print(match_string(string3))  # Output: False
print(match_string(string4))  # Output: False
print(match_string(string5))  # Output: False
```

Question 8- Write a Python program to find sequences of lowercase letters joined with a underscore.

Answer:

```
import re

def find_lowercase_sequences(input_string):
    pattern = r'[a-z]+_[a-z]+'
    lowercase_sequences = re.findall(pattern, input_string)
    return lowercase_sequences

# Example usage
```

```
input_text = "Hello_world is_a_common_naming_convention in_python_programming"

result = find_lowercase_sequences(input_text)

print(result)  # Output: ['is_a', 'common_naming_convention', 'in_python']
```

Question 9- Write a Python program that matches a string that has an 'a' followed by anything, ending in 'b'.

Answer:

```
import re


def match_string(input_string):
    pattern = r'^a.*b$'
    if re.match(pattern, input_string):
        return True
    else:
        return False


# Example usage
string1 = "aabcdb"
string2 = "aacde"
string3 = "abb"
string4 = "abbc"
string5 = "abcd"

print(match_string(string1))  # Output: True
print(match_string(string2))  # Output: False
print(match_string(string3))  # Output: True
print(match_string(string4))  # Output: True
print(match_string(string5))  # Output: False
```

Question 10- Write a Python program that matches a word at the beginning of a string.

Answer:

```
import re
```

```
def match_word_at_beginning(input_string, word):
    pattern = r'^' + word
    if re.match(pattern, input_string):
        return True
    else:
        return False

# Example usage
string1 = "Hello, world!"
string2 = "Python is awesome."
word1 = "Hello"
word2 = "Python"
word3 = "world"

print(match_word_at_beginning(string1, word1))  # Output: True
print(match_word_at_beginning(string1, word2))  # Output: False
print(match_word_at_beginning(string1, word3))  # Output: False

print(match_word_at_beginning(string2, word1))  # Output: False
print(match_word_at_beginning(string2, word2))  # Output: True
print(match_word_at_beginning(string2, word3))  # Output: False
```

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

Answer:

```
import re

def match_string(input_string):
    pattern = r'^[a-zA-Z0-9_]+$'
    if re.match(pattern, input_string):
        return True
```

```
    else:
        return False

# Example usage
string1 = "Hello123"
string2 = "Hello_World"
string3 = "Hello@123"
string4 = "Hello World"

print(match_string(string1))  # Output: True
print(match_string(string2))  # Output: True
print(match_string(string3))  # Output: False
print(match_string(string4))  # Output: False
```

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

Answer:

```
def starts_with_number(input_string, number):
    if input_string.startswith(str(number)):
        return True
    else:
        return False

# Example usage
string1 = "12345abc"
string2 = "98765xyz"
string3 = "abc123"

print(starts_with_number(string1, 123))  # Output: True
print(starts_with_number(string2, 987))  # Output: True
print(starts_with_number(string3, 123))  # Output: False
```

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

Answer: def remove_leading_zeros(ip_address):

```
    parts = ip_address.split(".")
    trimmed_parts = [str(int(part)) for part in parts]
    trimmed_ip_address = ".".join(trimmed_parts)
    return trimmed_ip_address

# Example usage
ip_address1 = "192.168.001.001"
ip_address2 = "10.0.01.01"
ip_address3 = "001.002.003.004"

print(remove_leading_zeros(ip_address1))  # Output: 192.168.1.1
print(remove_leading_zeros(ip_address2))  # Output: 10.0.1.1
print(remove_leading_zeros(ip_address3))  # Output: 1.2.3.4
```

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'.

Output- August 15th 1947

Hint- Use re.match() method here

Answer: import re

```
def match_date_string(file_path):
    with open(file_path, 'r') as file:
        text = file.read()
        pattern = r'[A-Z][a-z]+\s\d+(?:st|nd|rd|th)\s\d{4}'
        match = re.match(pattern, text)
        if match:
```

```
        return match.group()
    else:
        return None

# Example usage
file_path = 'sample_text.txt'
date_string = match_date_string(file_path)
print(date_string)  # Output: August 15th 1947
```

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

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

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

Answer:

```
def search_strings(text, searched_words):
  found_words = []
  for word in searched_words:
    if word in text:
      found_words.append(word)
  return found_words

# Example usage
text = 'The quick brown fox jumps over the lazy dog.'
searched_words = ['fox', 'dog', 'horse']

found_words = search_strings(text, searched_words)
print(found_words)  # Output: ['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'

Answer:

```
import re
```

```
def search_string_with_location(text, searched_word):
    pattern = re.escape(searched_word)
    matches = re.finditer(pattern, text)
    locations = [match.start() for match in matches]
    return locations

# Example usage
text = 'The quick brown fox jumps over the lazy dog.'
searched_word = 'fox'

match_locations = search_string_with_location(text, searched_word)
print(match_locations)  # Output: [16]
```

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

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

Pattern : 'exercises'.

Answer:

```
def find_substrings(text, pattern):
    substrings = []
    length = len(pattern)
    index = 0

    while index < len(text):
        index = text.find(pattern, index)
        if index == -1:
            break
        substrings.append(text[index:index+length])
        index += length

    return substrings
```

```
# Example usage
text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'

found_substrings = find_substrings(text, pattern)
print(found_substrings)  # Output: ['exercises', 'exercises', 'exercises']
```

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

Answer:

```
def find_substring_occurrences(text, pattern):
    occurrences = []
    index = 0

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

    return occurrences

# Example usage
text = 'Python exercises, PHP exercises, C# exercises'
pattern = 'exercises'

found_occurrences = find_substring_occurrences(text, pattern)
for occurrence in found_occurrences:
    substring, position = occurrence
```

```
    print(f"Substring: {substring} | Position: {position}")
```

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

Answer:

```
def convert_date_format(date):
    parts = date.split('-')
    converted_date = f"{parts[2]}-{parts[1]}-{parts[0]}"
    return converted_date

# Example usage
date = "2022-07-15"
converted_date = convert_date_format(date)
print(converted_date)  # Output: 15-07-2022
```

Question 20- Write a Python program to find all words starting with 'a' or 'e' in a given string.

Answer:

```
import re

def find_words_starting_with_a_or_e(input_string):
    pattern = r'\b[aAeE]\w+\b'
    words = re.findall(pattern, input_string)
    return words

# Example usage
input_text = "An apple is eaten by an elephant."
found_words = find_words_starting_with_a_or_e(input_text)
print(found_words)  # Output: ['An', 'apple', 'eaten', 'elephant']
```

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

Answer:

```
import re

def separate_numbers_with_positions(input_string):
    pattern = r'\d+'
```

```
    numbers = re.findall(pattern, input_string)

    positions = []
    for match in re.finditer(pattern, input_string):
        start = match.start()
        positions.append(start)

    return numbers, positions

# Example usage
input_text = "Hello 123, I am 456. How are you?"
numbers, positions = separate_numbers_with_positions(input_text)

for i in range(len(numbers)):
    number = numbers[i]
    position = positions[i]
    print(f"Number: {number} | Position: {position}")
```

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

Answer:

```
import re

def extract_maximum_numeric_value(input_string):
    pattern = r'\d+'
    numbers = re.findall(pattern, input_string)

    if numbers:
        max_number = max(map(int, numbers))
        return max_number
    else:
```

```
    return None


# Example usage
input_text = "The maximum number is 98765, but there are also 12345 and 54321."
max_number = extract_maximum_numeric_value(input_text)
print(max_number)  # Output: 98765
```

Question 23- Write a Regex in Python to put spaces between words starting with capital letters

Answer:

```
import re

def add_spaces_between_capital_words(input_string):
    pattern = r'(?<!^)(?=[A-Z])'
    modified_string = re.sub(pattern, ' ', input_string)
    return modified_string

# Example usage
input_text = "HelloWorld,HowAreYouToday?"
modified_text = add_spaces_between_capital_words(input_text)
print(modified_text)  # Output: "Hello World, How Are You Today?"
```

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

Answer:

```
import re

def find_sequences_of_uppercase_followed_by_lowercase(input_string):
    pattern = r'[A-Z][a-z]+'
    sequences = re.findall(pattern, input_string)
    return sequences

# Example usage
input_text = "The Quick Brown Fox Jumps Over The Lazy Dog"
found_sequences = find_sequences_of_uppercase_followed_by_lowercase(input_text)
```

print(found_sequences) # Output: ['Quick', 'Brown', 'Fox', 'Jumps', 'Over', 'The', 'Lazy', 'Dog']

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

Answer: import re

```
def remove_duplicate_words(sentence):
    pattern = r'\b(\w+)\b\s+(?=.*\b\1\b)'
    modified_sentence = re.sub(pattern, '', sentence)
    return modified_sentence.strip()

# Example usage
input_sentence = "This is is a test test sentence sentence."
modified_sentence = remove_duplicate_words(input_sentence)
print(modified_sentence)  # Output: "This is a test sentence."
```

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

Answer: import re

```
def is_string_ending_with_alphanumeric(input_string):
    pattern = r'^.*[a-zA-Z0-9]$'
    if re.match(pattern, input_string):
        return True
    else:
        return False

# Example usage
string1 = "Hello123"
string2 = "abc_"
string3 = "1234"
string4 = "Hello_World"
```

print(is_string_ending_with_alphanumeric(string1)) # Output: True

print(is_string_ending_with_alphanumeric(string2)) # Output: False

print(is_string_ending_with_alphanumeric(string3)) # Output: True

print(is_string_ending_with_alphanumeric(string4)) # Output: False

Question 27-Write a python program using RegEx to extract the hashtags.

Sample Text: 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"""

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

Answer: import re

def extract_hashtags(text):

  pattern = r'#\w+'

  hashtags = re.findall(pattern, text)

  return hashtags

 # Example usage

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'

hashtags = extract_hashtags(text)

print(hashtags) # Output: ['#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"

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

Answer: import re

```
def remove_u_plus_symbols(text):
    pattern = r'<U\+[A-Za-z0-9]{4}>'
    modified_text = re.sub(pattern, '', text)
    return modified_text

# Example usage
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"
modified_text = remove_u_plus_symbols(text)
print(modified_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.

Store this sample text in the file and then extract dates.

Answer: import re

```
def extract_dates_from_file(file_path):
    with open(file_path, 'r') as file:
        text = file.read()
        pattern = r'\d{2}-\d{2}-\d{4}'
        dates = re.findall(pattern, text)
        return dates

# Example usage
file_path = 'sample_text.txt'
dates = extract_dates_from_file(file_path)
print(dates)  # Output: ['12-09-1992', '15-12-1999']
```

Question 30- Write a Python program to replace all occurrences of a space, comma, or dot with a colon.

Sample Text- 'Python Exercises, PHP exercises.'

Output: Python:Exercises::PHP:exercises:

Answer:

```
def replace_with_colon(input_text):
    modified_text = input_text.replace(' ', ':').replace(',', ':').replace('.', ':')
    return modified_text

# Example usage
text = 'Python Exercises, PHP exercises.'
modified_text = replace_with_colon(text)
print(modified_text)  # Output: 'Python:Exercises::PHP:exercises:'
```