**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:

def replace\_chars(string):

chars=[' ',',','.']

new\_string= string

for char in chars:

new\_string= new\_string.replace(char,':')

return new\_string

string= 'Python Exercises, PHP exercises.'

result= replace\_chars(string)

print (result)

**Output:**   
**Python:Exercises::PHP:exercises:**

**Question 2-** Write a Python program to find all words starting with 'a' or 'e' in a given string.

def words\_starting\_with\_a\_or\_e(string):

words = re.findall(r'\b[ae]\w+', string)

return words

string = 'Mounisha is an Msc Finance graduate and also Mounisha is an MBA finance graduate.'

result = words\_starting\_with\_a\_or\_e(string)

print(result)

**Output:**

**['an', 'and', 'also', 'an']**

**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.

def long\_words(string):

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

words = rule.findall(input\_string)

return words

input\_string = "The fear of the Lord is the beginning of wisdom and knowledge of the Holy One is understanding"

result = long\_words(input\_string)

print(result)

**Output:**

**['beginning', 'wisdom', 'knowledge', 'understanding']**

**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.

def long\_words(string):

rule = re.compile(r'\b\w{3,6}\b')

words = rule.findall(input\_string)

return words

string = "The fear of the Lord is the beginning of wisdom and knowledge of the Holy One is understanding"

result = long\_words(string)

print(result)

**Output:**

**['The', 'fear', 'the', 'Lord', 'the', 'wisdom', 'and', 'the', 'Holy', 'One']**

**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

def parentheses\_words(strings):

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

stringss = []

for string in strings:

match = pattern.match(string)

if match:

first = match.group(1)

middle = match.group(2)

end = match.group(3)

stringss.append(f"{first}{middle}{end}")

else:

stringss.append(string)

return stringss

string = [

"example (.com)",

"hr@fliprobo (.com)",

"github (.com)",

"Hello (Data Science World)",

"Data (Scientist)"

]

result = parentheses\_words(string)

for string in result:

print(string)

**Output:**

**example.com**

**hr@fliprobo.com**

**github.com**

**HelloData Science World**

**DataScientist**

**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.

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

pattern = r'\s?\([^)]\*\)'

modified\_text = re.sub(pattern, '', sample\_text)

modified\_text\_list = re.findall(r'"([^"]\*)"', cleaned\_text)

print(modified\_text\_list)

**Output:**

**['example', 'hr@fliprobo', 'github', 'Hello', 'Data']**

**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’]

sample\_text = "ImportanceOfRegularExpressionsInPython"

words = re.findall(r'[A-Z][a-z]\*', sample\_text)

print(words)

**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

def spaces\_bw\_no(text):

modified\_text = re.sub(r'([A-Za-z])(\d)', r'\1 \2', text)

return modified\_text

sample\_text = "RegularExpression1IsAn2ImportantTopic3InPython"

result = spaces\_bw\_no(sample\_text)

print(result)

**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

def insert\_spaces(text):

spaced\_text = re.sub(r'(?<=[a-z])(?=[A-Z0-9])|(?<=\d)(?=\D)', ' ', text)

return spaced\_text

sample\_text = "RegularExpression1IsAn2ImportantTopic3InPython"

result = insert\_spaces(sample\_text)

print(result)

**Output:**

**Regular Expression 1 Is An 2 Important Topic 3 In Python**

**Question 10-** Write a python program to extract email address from the text stored in the text using Regular Expression.

**Sample Text-** Hello my name is Data Science and my email address is [xyz@domain.com](mailto:xyz@domain.com) and alternate email address is [xyz.abc@sdomain.domain.com](mailto:xyz.abc@sdomain.domain.com).

Please contact us at hr@fliprobo.com for further information.

**Expected Output:**

['xyz@domain.com', 'xyz.abc@sdomain.domain.com']

['hr@fliprobo.com']

**Note-** Store given sample text in the text file and then extract email addresses.

sample\_text = "Hello my name is Data Science and my email address is xyz@domain.com and alternate email address is xyz.abc@sdomain.domain.com. Please contact us at hr@fliprobo.com for further information."

file\_path = "sample\_text\_file.txt"

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

file.write(sample\_text)

def extract\_email\_addresses\_from\_text\_file(file\_path):

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

content = file.read()

email\_pattern = r"\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b"

email\_addresses = re.findall(email\_pattern, content)

return email\_addresses

extracted\_email\_addresses = extract\_email\_addresses\_from\_text\_file(file\_path)

for email in extracted\_email\_addresses:

print(email)

**Output:**

**xyz@domain.com**

**xyz.abc@sdomain.domain.com**

**hr@fliprobo.com**

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

def is\_valid\_string(input\_string):

pattern = r'^[A-Za-z0-9\_]\*$'

return re.match(pattern, input\_string) is not None

test\_strings = [

"Mouni\_0809",

"Mounisha\_Srinivaseelu\_Naidu",

"mounisha!",

"Mounisha Srinivasan",

]

for test\_string in test\_strings:

if is\_valid\_string(test\_string):

print(f'"{test\_string}" is valid.')

else:

print(f'"{test\_string}" is invalid.')

**Output:**

**"Mouni\_0809" is valid.**

**"Mounisha\_Srinivaseelu\_Naidu" is valid.**

**"mounisha!" is invalid.**

**"Mounisha Srinivasan" is invalid.**

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

def number(input\_string, target\_number):

return input\_string.startswith(str(target\_number))

test\_strings = ["123mouni", "456mouni", "789mouni"]

target\_number = 456

for string in test\_strings:

if number(string, target\_number):

print(f'"{string}" starts with numbers {target\_number}.')

else:

print(f'"{string}" does not start with numbers {target\_number}.')

**Output:**

**"123mouni" does not start with numbers 456.**

**"456mouni" starts with numbers 456.**

**"789mouni" does not start with numbers 456.**

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

def remove\_zeros(ip\_address):

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

normalized\_parts = [str(int(part)) for part in parts]

return '.'.join(normalized\_parts)

test\_ip\_addresses = ["192.168.011.001", "010.220.030.040", "020.000.000.000"]

for ip in test\_ip\_addresses:

normalized\_ip = remove\_zeros(ip)

print(f'Original IP: {ip}, Modified IP: {normalized\_ip}')

**Output:**

**Original IP: 192.168.011.001, Modified IP: 192.168.11.1**

**Original IP: 010.220.030.040, Modified IP: 10.220.30.40**

**Original IP: 020.000.000.000, Modified IP: 20.0.0.0**

**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.

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."

file\_path = "sample\_text\_file.txt"

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

file.write(sample\_text)

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

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

content = file.read()

date\_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"

date\_match = re.search(date\_pattern, content)

if date\_match:

return date\_match.group()

else:

return "Date not found."

extracted\_date = extract\_date\_from\_text\_file(file\_path)

print("Extracted Date:")

print(extracted\_date)

**Output:**

**Extracted Date:**

**August 15th 1947**

**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'

def words(text, words):

result\_words = [word for word in words if word in text]

return result\_words

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

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

result\_words = words(sample\_text, searched\_words)

for word in result\_words:

print(f'Found: {word}')

for word in searched\_words:

if word not in result\_words:

print(f'Not Found: {word}')

**Output:**

**Found: fox**

**Found: dog**

**Not Found: horse**

**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'

def search\_and\_location(text, pattern):

find\_locations = []

start = 0

while start < len(text):

pos = text.find(pattern, start)

if pos == -1:

break

find\_locations.append(pos)

start = pos + 1

return find\_locations

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

searched\_word = 'fox'

find\_locations = search\_and\_location(sample\_text, searched\_word)

if find\_locations:

print(f'Wanted\_word "{searched\_word}" positions: {find\_locations}')

else:

print(f'Wanted\_word "{searched\_word}" not found.')

**Output:**

**Wanted\_word "fox" 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'.

def find\_substrings(text, pattern):

matches = re.findall(pattern, text)

return matches

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

search\_pattern = 'exercises'

substrings = find\_substrings(sample\_text, search\_pattern)

print(f"Occurrences of '{search\_pattern}': {substrings}")

**Output:**

**Occurrences of 'exercises': ['exercises', 'exercises', 'exercises']**

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

def find\_substring(main\_string, substring):

occurrences = []

position = -1

while True:

position = main\_string.find(substring, position + 1)

if position == -1:

break

occurrences.append(position)

return occurrences

main\_string = "Mounisha is an Msc Finance graduate and also Mounisha is an MBA finance graduate"

substring = "Mounisha"

occurrences = find\_substring(main\_string, substring)

print(f"Occurrence '{substring}' in '{main\_string}': {len(occurrences)} ")

print("Positions:", occurrences)

**Output:**

**Enter the course: MBA IN FINANCE**

**Enter the subject to search: FINANCE**

**Substring 'FINANCE' positions: [7]**

**Total occurrences: 1**

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

def convert\_date(date):

parts = date.split('-')

if len(parts) != 3:

return "Invalid date format"

year, month, day = parts

converted\_date = f"{day}-{month}-{year}"

return converted\_date

input\_date = "1999-09-09"

output\_date = convert\_date(input\_date)

print(output\_date)

**Output:**

**Enter date in yyyy-mm-dd format: 1999-09-09**

**Converted date: 09-09-1999**

**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']

def decimal\_no(text):

pattern = re.compile(r'\b\d+\.\d{1,2}\b')

decimal\_numbers = pattern.findall(text)

return decimal\_numbers

sample\_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"

result = decimal\_no(sample\_text)

print(result)

**Output:**

**['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.

def extract\_numbers\_and\_positions(input\_string):

numbers\_with\_positions = []

start = 0

while start < len(input\_string):

while start < len(input\_string) and not input\_string[start].isdigit():

start += 1

if start < len(input\_string):

end = start

while end < len(input\_string) and input\_string[end].isdigit():

end += 1

number = input\_string[start:end]

numbers\_with\_positions.append((number, start))

start = end

return numbers\_with\_positions

sample\_string = "Mounisha graduate in 1999"

numbers\_and\_positions = extract\_numbers\_and\_positions(sample\_string)

for number, position in numbers\_and\_positions:

print(f"Number: {number}, Position: {position}")

**Output:**

**Number: 1999, Position: 21**

**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

def extract\_max(text):

numbers = re.findall(r'\d+', text)

if numbers:

max\_number = max(map(int, numbers))

return max\_number

else:

return None

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

max\_value = extract\_max(sample\_text)

if max\_value is not None:

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

else:

print("No numeric values found in the text.")

**Output:**

**Maximum 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

def insert\_spaces(input\_string):

words = re.findall(r'[A-Z][a-z]\*', input\_string)

formatted\_string = ' '.join(words)

return formatted\_string

sample\_text = "RegularExpressionIsAnImportantTopicInPython"

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

print(modified\_text)

**Output:**

**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

sample\_text = "Mounisha is an Msc Finance graduate and also Mounisha is an MBA finance graduate"

matches = re.findall(r'[A-Z][a-z]\*', sample\_text)

print(matches)

**Output:**

**['Mounisha', 'Msc', 'Finance', 'Mounisha', 'M', 'B', 'A']**

**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

def continuous\_duplicates(sentence):

modified\_sentence = re.sub(r'\b(\w+)(?:\s+\1)+\b', r'\1', sentence)

return modified\_sentence

sample\_text = "Hello hello world world"

modified\_text = continuous\_duplicates(sample\_text)

print(modified\_text)

**Output:**

**Hello hello world**

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

def is\_valid\_string(input\_string):

pattern = re.compile(r'^.\*[a-zA-Z0-9]$')

if pattern.match(input\_string):

return True

return False

sample\_strings = ["mouni123", "mouni456", "mouni#"]

for string in sample\_strings:

if is\_valid\_string(string):

print(f"'{string}' is valid.")

else:

print(f"'{string}' is not valid.")

**Output:**

**'mouni123' is valid.**

**'mouni456' is valid.**

**'mouni#' is not valid.**

**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']

new\_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"""

pattern = r"#\w+"

extract\_hashtags = re.findall(pattern, sample\_text)

print(extract\_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"

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

new\_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"

pattern = r"<U\+[A-F0-9]+>"

remove\_symbols = re.sub(pattern, "", new\_text)

print(remove\_symbols)

**Output:**

**@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.

sample\_text = "Ron was born on 12-09-1992 and he was admitted to school 15-12-1999."

file\_path = "sample\_text\_file.txt"

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

file.write(sample\_text)

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

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

content = file.read()

date\_pattern = r"\d{2}-\d{2}-\d{4}"

dates = re.findall(date\_pattern, content)

return dates

extracted\_dates = extract\_dates\_from\_file(file\_path)

print("Extracted Dates:")

for date in extracted\_dates:

print(date)

**Output:**

**Extracted Dates:**

**12-09-1992**

**15-12-1999**

**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.

def remove\_words(text):

pattern = re.compile(r'\b\w{2,4}\b')

new\_text = pattern.sub('', text)

return new\_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\_words(sample\_text)

print(result)

**Output:**

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