Internship Assignment RegEx

December 16, 2023

[1]:
```
#1. Write a Python program to replace all occurrences of a space, comma, or dot with a colon.
```

[4]:
```
import pandas as pd
import re
import regex as re
```

[7]:
```
sample_text = 'Python Exercises, PHP exercises.'
output = re.sub("[\s,\.]",":",sample_text)
print(output)
```

```
Python:Exercises::PHP:exercises:
```

[ ]:
```
```

[8]:
```
#2. Create a dataframe using the dictionary below and remove everything (commas (,), !, XXXX, ;, etc.) from the columns except words.
```

[13]:
```
dict1= {'SUMMARY' : ['hello, world!', 'XXXXX test', '123four, five:; six...']}
dict1
```

[13]:
```
{'SUMMARY': ['hello, world!', 'XXXXX test', '123four, five:; six…']}
```

[21]:
```
df1=pd.DataFrame(dict1)
```

[22]:
```
print(df1)
```

```
                  SUMMARY
0           hello, world!
1              XXXXX test
2  123four, five:; six…
```

[28]:
```
df1['SUMMARY'].str.replace('[,!;:\.^X\d]',"",regex=True)
```

[28]:
```
0      hello world
1             test
2    four five six
Name: SUMMARY, dtype: object
```

[ ]:

[ ]:
```
#5. Create a function in Python to remove the parenthesis in a list of strings.
 ↪The use of the re.compile() method is mandatory.
```

[13]:
```
import re
```

[14]:
```
list1=["example (.com)", "hr@fliprobo (.com)", "github (.com)", "Hello (Data
 ↪Science World)", "Data (Scientist)"]
list1
```

[14]:
```
['example (.com)',
 'hr@fliprobo (.com)',
 'github (.com)',
 'Hello (Data Science World)',
 'Data (Scientist)']
```

[15]:
```
def remove_parenthesis(strings):
    pattern=re.compile(r'[()]')
    result=[pattern.sub('',s) for s in strings]
    return result
```

[16]:
```
expected_output=remove_parenthesis(list1)
for i in expected_output:
    print(i)
```

```
example .com
hr@fliprobo .com
github .com
Hello Data Science World
Data Scientist
```

[ ]:

[ ]:
```
#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.
```

[72]:
```
import re
```

[73]:
```
string1= "Improve your productivity by learning Generative AI tools and stand
 ↪out as a Generative AI enabled data professional."
```

[74]:
```
#Print result for all "atleast" 4 characters long in a string
def find_atleast(string):
    pattern1=re.compile(r"\w{4,}")
    result=pattern1.findall(string)
    return result
```

```
final_result=find_atleast(string1)
print(final_result)
```

```
['Improve', 'your', 'productivity', 'learning', 'Generative', 'tools', 'stand',
'Generative', 'enabled', 'data', 'professional']
```

[75]:
```
#Print result for "only 4" characters long in a string
def find_only(string):
    pattern2=re.compile(r"\w{4}")
    result2=pattern2.findall(string)
    return result2

final_result2=find_only(string1)
print(final_result2)
```

```
['Impr', 'your', 'prod', 'ucti', 'vity', 'lear', 'ning', 'Gene', 'rati', 'tool',
'stan', 'Gene', 'rati', 'enab', 'data', 'prof', 'essi', 'onal']
```

[ ]:

[ ]:
```
#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.
```

[79]:
```
import re
```

[80]:
```
string2="Current approaches to NLP are based on machine learning. NLP tasks␣
 ↪include text translation and speech recognition."
```

[81]:
```
def find_345(string):
    pattern_match=re.compile(r"\w{3,5}")
    result3=pattern_match.findall(string)
    return result3

final_result3=find_345(string2)
print(final_result3)
```

```
['Curre', 'appro', 'aches', 'NLP', 'are', 'based', 'machi', 'learn', 'ing',
'NLP', 'tasks', 'inclu', 'text', 'trans', 'latio', 'and', 'speec', 'recog',
'nitio']
```

[ ]:

[ ]:
```
#6. Write a python program to remove the parenthesis area from the text stored␣
 ↪in the text file using Regular Expression
```

[115]:
```
import re
```

[116]:
```
#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)"]

text_file1 = "sample_text_file.txt"

with open(text_file1, "w") as file:
    for line in sample_text:
        file.write(line + '\n')
```

[117]:
```
with open(text_file1, "r") as file:
    textfrom_file=file.readlines()
    result=[re.sub(r'\s*\([^)]*\)\s*',"",line) for line in textfrom_file]

print(result)
```

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

[ ]:

[ ]:
```
#7. Write a regular expression in Python to split a string into uppercase␣
↪letters.
```

[122]:
```
import re
sample_text = "ImportanceOfRegularExpressionsInPython"
```

[121]:
```
result = re.findall(r'[A-Z][^A-Z]*',sample_text)

print(result)
```

```
['Importance', 'Of', 'Regular', 'Expressions', 'In', 'Python']
```

[ ]:

[ ]:
```
#8. Create a function in python to insert spaces between words starting with␣
↪numbers.
```

[147]:
```
import re
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
```

[148]:
```
def insert_spaces(string):
    result = re.sub(r'([A-Za-z])(\d)',r'\1 \2',string)
    return result
```

```
desired_output=insert_spaces(sample_text)
print(desired_output)
```

```
RegularExpression 1IsAn 2ImportantTopic 3InPython
```

[ ]:

[ ]:
```
#9. Create a function in python to insert spaces between words starting with␣
↪capital letters or with numbers.
```

[167]:
```
import re
sample_text = "RegularExpression1IsAn2ImportantTopic3InPython"
```

[168]:
```
def insert_spaces2(string):
    result=re.sub(r'([A-Za-z])(\d)',r'\1 \2 ', string)
    return result

desired_output2=insert_spaces2(sample_text)
print(desired_output2)
```

```
RegularExpression 1 IsAn 2 ImportantTopic 3 InPython
```

[ ]:

[ ]:
```
#11. Write a Python program to match a string that contains only upper and␣
↪lowercase letters, numbers, and underscores.
```

[78]:
```
import re
```

[81]:
```
def match_string(string):
    pattern=re.compile(r'^[a-zA-Z0-9_]+$')

    if pattern.match(string):
        return True
    else:
        return False


testing_string="ABC123_abc"
if match_string(testing_string):
    print("Valid String")
    print(testing_string)
else:
    print("Invalid String")
```

```
Valid String
ABC123_abc
```

[ ]:

[82]:
```
#12. Write a Python program where a string will start with a specific number
```

[97]:
```
import re
```

[106]:
```
def specific_number(string,specified_number):
    pattern=re.compile(fr'^{specified_number}')
    if pattern.match(string):
        print(f"The string: '{string}' starts with {specified_number}")
        return True
    else:
        print(f"The string: '{string}' does not start with {specified_number}")
        return False


testing_string="100 is the best score"
specified_number ="100"
specific_number(testing_string,specified_number)
```

```
The string: '100 is the best score' starts with 100
```

[106]:
```
True
```

[ ]:

[ ]:
```
#13. Write a Python program to remove leading zeros from an IP address
```

[114]:
```
import re
def remove_zeros(ip_address):
    pattern=re.compile(r'\b0+(\d+)\b')

    cleaned_ip=pattern.sub(r'\1',ip_address)
    return cleaned_ip


ip_address_withzeros="100.345.005.078"
cleaned_ipaddress= remove_zeros(ip_address_withzeros)
print(f"IP Address: {ip_address_withzeros}")
print(f"Cleaned IP Address: {cleaned_ipaddress}")
```

```
IP Address: 100.345.005.078
Cleaned IP Address: 100.345.5.78
```

[ ]:

[ ]:
```
#30. Create a function in python to remove all words from a string of length␣
 ↪between 2 and 4.
```

[148]:
```
import re
```

[149]:
```
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."
```

[150]:
```
def remove_words(string):
    pattern= re.compile(r'\b\w{2,4}\b\s*')
    result=pattern.sub("",string)
    return result

final_output=remove_words(sample_text)
print (final_output)
```

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

[ ]:

[167]:
```
#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.
```

[190]:
```
import re
```

[191]:
```
sample_text="Ron was born on 12-09-1992 and he was admitted to school␣
 ↪15-12-1999."
```

[192]:
```
text_file2 = "date_text_file.txt"

with open(text_file2, "w") as file:
    for line in sample_text:
        file.write(line)
```

[193]:
```
def extract_dates(file_path):
    with open(file_path, "r") as file:
        textfrom_file=file.read()
        date_pattern=re.compile(r'\b\d{2}-\d{2}-\d{4}\b')
        dates=date_pattern.findall(textfrom_file)
        return dates
```

```
file_path= "date_text_file.txt"


output_dates=extract_dates(file_path)

if output_dates:
    print("Dates found in the text:")
    for date in output_dates:
        print(date)
else:
    print("No dates found")
```

```
Dates found in the text:
12-09-1992
15-12-1999
```

[ ]:

[ ]:
```
#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.
```

[197]:
```
import re

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"
```

[198]:
```
def remove_symbols(input_string):
    removing_pattern=re.compile(r'<U\+\w{4}>')
    expected_output = removing_pattern.sub("",input_string)
    return expected_output

output= remove_symbols(sample_text)
print(output)
```

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

[ ]:

[ ]:
```
#27. Write a python program using RegEx to extract the hashtags.
```

[204]:
```
import re
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"""
```

[205]:
```
output = re.findall(r'#\w+',sample_text)
print(output)
```

```
['#Doltiwal', '#xyzabc', '#Demonetization']
```

[ ]:

[ ]:
```
#26. Write a python program using RegEx to accept string ending with␣
 ↪alphanumeric character.
```

[229]:
```
import re

def ends_with_alphanum(string):
    pattern = re.compile(r'.*\w$')
    return bool(pattern.match(string))

test_string = " This is test Scientist123_5"
result = ends_with_alphanum(test_string)

if result:
    print(f"The string '{test_string}' ends with an alphanumeric character.")
else:
    print(f"The string '{test_string}' does not end with an alphanumeric␣
 ↪character.")
```

```
The string ' This is test Scientist123_5' ends with an alphanumeric character.
```

[ ]:

[1]:
```
#25. Write a Python program to remove continuous duplicate words from Sentence␣
 ↪using Regular Expression.
```

[20]:
```
import re
sample_text = "Hello hello world world"
```

[21]:
```
def remove_duplicate(sentence):
    pattern = re.compile(r'\b(\w+)(?:\s+\1\b)+')
    result = pattern.sub(r'\1',sentence)
    return result


desired_result = remove_duplicate(sample_text)
print("New Desired Sentence:")
```

```
print(desired_result)
```

```
New Desired Sentence:
Hello hello world
```

[ ]:

[31]:
```
#24. Python regex to find sequences of one upper case letter followed by lower␣
↪case letters.
```

[32]:
```
import re
```

[33]:
```
def find_sequences(input_text):
    pattern = re.compile(r'[A-Z][a-z]+')
    result = pattern.findall(input_text)
    return result

sample_text = "Information technology (IT) is a set of related fields that␣
↪encompass Computer Systems, Software, Programming."
desired_sequence = find_sequences(sample_text)
print(desired_sequence)
```

```
['Information', 'Computer', 'Systems', 'Software', 'Programming']
```

[ ]:

[ ]:
```
#23. Create a function in python to insert spaces between words starting with␣
↪capital letters.
```

[50]:
```
import re
sample_text =  "RegularExpressionIsAnImportantTopicInPython"
```

[52]:
```
def insert_spaces(input_string):
    pattern = re.sub(r'(?<=[a-z])(?=[A-Z])'," ", input_string)
    return pattern

result = insert_spaces(sample_text)
print("The original text:")
print(sample_text)
print("The desired text:")
print(result)
```

```
The original text:
RegularExpressionIsAnImportantTopicInPython
The desired text:
Regular Expression Is An Important Topic In Python
```

[ ]:

[53]:
```
#22. Write a regular expression in python program to extract maximum/largest␣
↪numeric value from a string.
```

[70]:
```
import re
sample_text = 'My marks in each semester are: 947, 896, 926, 524, 734, 950, 642'
```

[71]:
```
def find_maximum(input_text):
    numbers = [int(match) for match in re.findall(r'\b\d+\b',input_text)]
    if numbers:
        return max(numbers)
    else:
        return none

desired_output= find_maximum(sample_text)
print(desired_output)
```

```
950
```

[ ]:

[73]:
```
#21. Write a Python program to separate and print the numbers and their␣
↪position of a given string.
```

[86]:
```
import re
```

[87]:
```
def findnumber_position(input_text):
    pattern = re.finditer(r'\b\d+\b',input_text)
    for match in pattern:
        number = match.group()
        position = match.start()
        print(f"The number: {number} and its position is: {position}")

sample_text = "The data scientist in year 2023 can earn more than 50000 in 1␣
↪month and minimum salary is 15 Lacs in a year"
findnumber_position(sample_text)
```

```
The number: 2023 and its position is: 27
The number: 50000 and its position is: 51
The number: 1 and its position is: 60
The number: 15 and its position is: 90
```

[ ]:

[ ]:
```
#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.
```

[7]:
```
import re
sample_text = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"
```

[8]:
```
def find_allDecimal(input_string):
    pattern = re.compile(r'\b\d+\.\d{1,2}\b')
    desired_output = pattern.findall(input_string)
    return desired_output

result = find_allDecimal(sample_text)
print(result)
```

```
['01.12', '145.8', '3.01', '27.25', '0.25']
```

[ ]:

[9]:
```
#19. Write a Python program to convert a date of yyyy-mm-dd format to␣
 ↪dd-mm-yyyy format.
```

[40]:
```
import re
```

[41]:
```
def convert_date_format(date_string):
    pattern = re.compile(r'(\d{4})-(\d{1,2})-(\d{1,2})')

    match = pattern.match(date_string)
    if match:
        year,month,day = match.groups()
        expected_format= f'{day}-{month}-{year}'
        return expected_format
    else:
        return "Invalid date format"

sample_string = "2023-12-14"
newdate_format = convert_date_format(sample_string)
print (newdate_format)
```

```
14-12-2023
```

[ ]:

[ ]:
```
#17. Write a Python program to find the substrings within a string.
```

[1]:
```
import re
sample_text ='Python exercises, PHP exercises, C# exercises'
```

[3]:
```
def find_substrings(input_string):
    pattern = 'exercises'
    match = re.findall(pattern,input_string)
    return match
```

```
result = find_substrings(sample_text)
print("Match Substrings:", result)
```

```
Match Substrings: ['exercises', 'exercises', 'exercises']
```

[ ]:

[ ]:
```
#18. Write a Python program to find the occurrence and position of the␣
 ↪substrings within a string.
```

[13]:
```
import re
sample_text ='Python exercises, PHP exercises, C# exercises'
```

[14]:
```
def find_substrings_position(input_string):
    pattern = 'exercises'
    matches = re.finditer(pattern,input_string)

    print("Occurence and position of '{}':".format(pattern))
    for match in matches:
        start_position =match.start()
        end_position = match.end()
        print("Found at position {} to {}".format(start_position,end_position))


find_substrings_position(sample_text)
```

```
Occurence and position of 'exercises':
Found at position 7 to 16
Found at position 22 to 31
Found at position 36 to 45
```

[ ]:

[22]:
```
#15. Write a Python program to search some literals strings in a string.
```

[23]:
```
import re
sample_text = 'The quick brown fox jumps over the lazy dog.'
```

[24]:
```
def search_literals(input_string):
    searched_words = {'fox','dog','horse'}

    for word in searched_words:
        if re.search(word,input_string):
            print("{} found in the string".format(word))
        else:
            print("{} not found in the string".format(word))
```

```
search_literals(sample_text)
```

```
dog found in the string
horse not found in the string
fox found in the string
```

[ ]:

[ ]:
```
#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.'
```

[52]:
```
import re
sample_text ='The quick brown fox jumps over the lazy dog.'
```

[53]:
```
def search_literal_position(input_string):
    searched_literal ='fox'
    match = re.search(searched_literal,input_string)

    if match:
        start_pos = match.start()
        end_pos = match.end()
        print("Occurence of '{}' is found at position {} to {}".
 ↪format(searched_literal, start_pos, end_pos))
    else:
        print("'{}' not found in the string".format(searched_literal))



search_literal_position(sample_text)
```

```
Occurence of 'fox' is found at position 16 to 19
```

[ ]:

[96]:
```
#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.
```

[97]:
```
import re
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"
```

[98]:
```
def find_datestring(input_string):
    text_file3 = "date_text_file3.txt"

    with open(text_file3, "w") as file:
```

```
        for line in input_string:
            file.write(line)

    with open(text_file3, "r") as file:
        textfrom_file3=file.read()

        date_pattern=re.compile(r'([A-Za-z]+ \d{1,2}(?:st|nd|th|rd) \d{4})')
        match = date_pattern.findall(textfrom_file3)

        if match:
            return match[0]

        else:
            return "No matching date found"

result = find_datestring(sample_text)
print(result)
```

```
August 15th 1947
```

[ ]:

[ ]:
```
#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.
```

[12]:
```
import pandas as pd
import re
import requests
from io import StringIO

url= "https://raw.githubusercontent.com/dsrscientist/DSData/master/
 ↪happiness_score_dataset.csv"
response = requests.get(url)
data = StringIO(response.text)

df = pd.read_csv(data)

df['first_five_letters'] = df['Country'].apply(lambda x: re.
 ↪match(r'^[a-zA-Z]{1,6}',x).group())
df
```

[12]:

	Country	Region	Happiness Rank	\
0	Switzerland	Western Europe	1	
1	Iceland	Western Europe	2	

```
2        Denmark                   Western Europe                3
3         Norway                   Western Europe                4
4         Canada                    North America                5
..           …                               …              …
153       Rwanda               Sub-Saharan Africa             154
154        Benin               Sub-Saharan Africa             155
155        Syria  Middle East and Northern Africa             156
156      Burundi               Sub-Saharan Africa             157
157         Togo               Sub-Saharan Africa             158

     Happiness Score  Standard Error  Economy (GDP per Capita)   Family  \
0              7.587         0.03411                   1.39651  1.34951
1              7.561         0.04884                   1.30232  1.40223
2              7.527         0.03328                   1.32548  1.36058
3              7.522         0.03880                   1.45900  1.33095
4              7.427         0.03553                   1.32629  1.32261
..               …               …                         …       …
153            3.465         0.03464                   0.22208  0.77370
154            3.340         0.03656                   0.28665  0.35386
155            3.006         0.05015                   0.66320  0.47489
156            2.905         0.08658                   0.01530  0.41587
157            2.839         0.06727                   0.20868  0.13995

     Health (Life Expectancy)  Freedom  Trust (Government Corruption)  \
0                     0.94143  0.66557                        0.41978
1                     0.94784  0.62877                        0.14145
2                     0.87464  0.64938                        0.48357
3                     0.88521  0.66973                        0.36503
4                     0.90563  0.63297                        0.32957
..                        …      …                              …
153                   0.42864  0.59201                        0.55191
154                   0.31910  0.48450                        0.08010
155                   0.72193  0.15684                        0.18906
156                   0.22396  0.11850                        0.10062
157                   0.28443  0.36453                        0.10731

     Generosity  Dystopia Residual first_five_letters
0       0.29678            2.51738             Switze
1       0.43630            2.70201             Icelan
2       0.34139            2.49204             Denmar
3       0.34699            2.46531             Norway
4       0.45811            2.45176             Canada
..          …                  …                  …
153     0.22628            0.67042             Rwanda
154     0.18260            1.63328              Benin
155     0.47179            0.32858              Syria
156     0.19727            1.83302             Burund
```

```
157     0.16681            1.56726                Togo

[158 rows x 13 columns]
```

[ ]:

[ ]:

[ ]: