```
In [2]: import regex as re
```

```
In [5]: # to replace all occurrences of a space, comma or dot with a colon.
        sub_ = 'Python Exercises, PHP exercises.'
        x = re.sub("[ ,.]", ":", sub_)
        print(x)
```

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

```
In [98]: import pandas as pd
         # to create a dataframe and remove everything ( commas(,), !, XXXXX, ;,etc.) from the columns except words.
         string_ = {'SUMMARY': ['hello, world!', 'XXXXXtest', '123four, five:; six...']}
         df = pd.DataFrame(string_)
         df['SUMMARY'] = df['SUMMARY'].str.replace('[,|!|XXXXX|;|:|.|\d]', '', regex=True)
         print(df)
```

```
         SUMMARY
0    hello world
1           test
2  four five six
```

```
In [37]: # to find all words that are at least 4 characters.
         target_string = "find all words that are at least 4 characters long in a string."
         pattern = re.compile(r'\b\w{4,}\b')
         result = pattern.findall(target_string)
         print(result)
```

```
['find', 'words', 'that', 'least', 'characters', 'long', 'string']
```

```
In [28]: # find all three, four and five character words.
         target_string = "Create a function in Python to find all three, four, and five character words in a string."
         pattern = re.compile(r'\b\w{3,5}\b')
         result = pattern.findall(target_string)
         print(result)
```

```
['find', 'all', 'three', 'four', 'and', 'five', 'words']
```

```
In [13]: # to remove the parenthesis in a list of strings.
         strings = ["example(.com)", "hr@fliprobo(.com)", "github(.com)", "Hello (Data Science World)", "Data (Scientist)"]
         pattern = re.compile(r'[()]')
         for string in strings:
             modified_string = re.sub(pattern, "", string)
             print(modified_string)
```

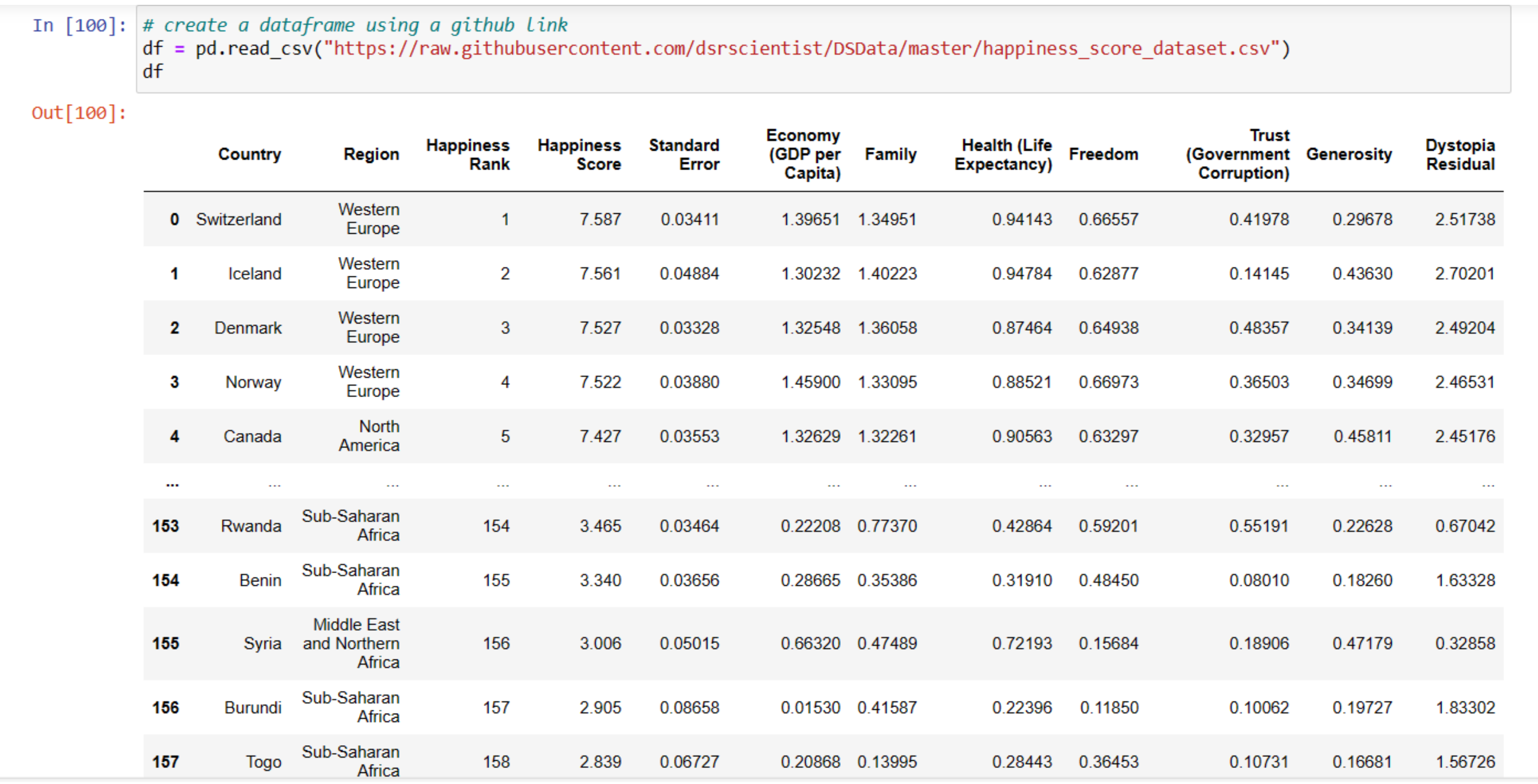
```
example.com
hr@fliprobo.com
github.com
Hello Data Science World
Data Scientist
```

```
In [22]: # remove the parenthesis area from the text stored in the text file.
         with open('examp.txt', 'r') as file:
             string = file.read()
             modified_string = re.sub(r'\([^()]*\)', '', string)
             print(modified_string)

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

```
In [9]: # split a string into uppercase letters.
        string = "ImportanceOfRegularExpressionInPython"
        pattern = '[A-Z][^A-Z]*'
        result = re.findall(pattern, string)
        print(result)

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

```
In [56]: # insert spaces between words starting with numbers.
         string = "RegularExpression1IsAn2ImportantTopic3InPython"
         pattern = r'(\d.)'
         result = re.sub(pattern, r' \1', string)
         print(result)

         RegularExpression 1IsAn 2ImportantTopic 3InPython
```

```
In [75]: # to insert spaces between words starting with capital letters or with numbers.
         string = "RegularExpression1IsAn2ImportantTopic3InPython"
         pattern = r'([A-Z][a-z]|\d+)'
         result = re.sub(pattern, r' \1', string)
         print(result)

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



```
In [100]: # create a dataframe using a github link
          df = pd.read_csv("https://raw.githubusercontent.com/dsrscientist/DSData/master/happiness_score_dataset.csv")
          df
```

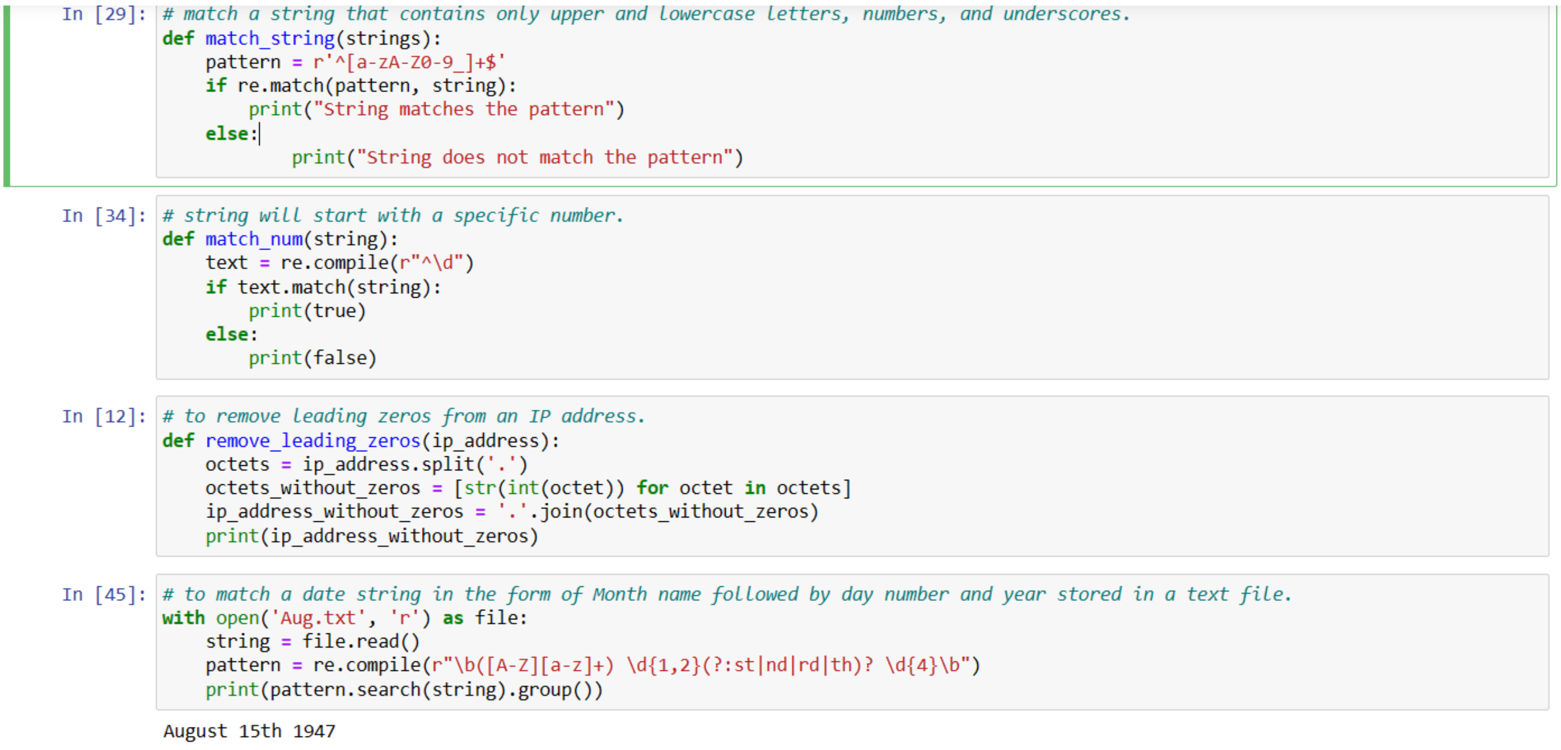
Out[100]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176
...	...	...	...	...	...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0.22628	0.67042
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260	1.63328
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47179	0.32858
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727	1.83302
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0.16681	1.56726

155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47179	0.32858
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727	1.83302
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0.16681	1.56726

158 rows × 12 columns

```
In [101]: pattern = r'\s(\w{6})'
          names = df['Country'].str.extract(pattern, expand=False)
          first_five_letters = names.value_counts()
```



```
In [29]: # match a string that contains only upper and lowercase letters, numbers, and underscores.
         def match_string(strings):
             pattern = r'^[a-zA-Z0-9_]+$'
             if re.match(pattern, string):
                 print("String matches the pattern")
             else:
                     print("String does not match the pattern")
```

```
In [34]: # string will start with a specific number.
         def match_num(string):
             text = re.compile(r"^\d")
             if text.match(string):
                 print(true)
             else:
                 print(false)
```

```
In [12]: # to remove leading zeros from an IP address.
         def remove_leading_zeros(ip_address):
             octets = ip_address.split('.')
             octets_without_zeros = [str(int(octet)) for octet in octets]
             ip_address_without_zeros = '.'.join(octets_without_zeros)
             print(ip_address_without_zeros)
```

```
In [45]: # to match a date string in the form of Month name followed by day number and year stored in a text file.
         with open('Aug.txt', 'r') as file:
             string = file.read()
             pattern = re.compile(r"\b([A-Z][a-z]+) \d{1,2}(?:st|nd|rd|th)? \d{4}\b")
             print(pattern.search(string).group())

         August 15th 1947
```

```
In [42]: # to serach some literals strings in a string.
         string = "The quick brown fox jumps over the lazy dog."
         pattern = "fox|dog|horse"
         result = re.findall(pattern, string)
         print(result)
```

```
['fox', 'dog']
```

```
In [7]: #  to search a literals string in a string and also find the location within the original string where the pattern occurs.
        string = "The quick brown fox jumps over the lazy dog."
        for msg in string:
            search = re.search('fox', string)
        print(search)
        print(search.group())
```

```
<regex.Match object; span=(16, 19), match='fox'>
fox
```

```
In [2]: # to find the substrings with a string.
        string = "Python exercises, PHP exercises, C# exercises"
        pattern = "exercises"
        result = re.findall(pattern, string)
        print(result)
```

```
['exercises', 'exercises', 'exercises']
```

```
In [8]: # to find the occurrence and position of the substrings within a string.
        string = "Pyhton exercises, PHP exercises, C# exercises"
        pattern = "exercises"
        for match in re.finditer(pattern, string):
            s = match.start()
            e = match.end()
            print('Found "%s" at %d:%d' % (string[s:e], s, e))
```

```
Found "exercises" at 7:16
Found "exercises" at 22:31
Found "exercises" at 36:45
```

```
In [11]: # to convert a date of yyyy-mm-dd from to dd-mm-yyyy format.
         date = "2024-03-05"
         x = re.sub(r"(\d{4})-(\d{1,2})-(\d{1,2})", "\\3-\\2-\\1", date)
         print(x)
```

```
05-03-2024
```

```
In [27]: # to find all decimal numbers with a precision of 1 or 2 in a string.
         string = "01.12 0132.123 2.31875 145.8 3.01 27.25 0.25"
         pattern = re.compile(r'\d+\.\d{1,2}\b')
         decimal_numbers = re.findall(pattern, string)
         print(decimal_numbers)
```

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

```
In [20]: # to seprate and print the numbers and their position of a given string.
         string = "Read the migration plan to Notebook 7 to learn about the new features."
         for m in re.finditer("\d+", string):
             print(m.group(0))
             print("Position:", m.start())

         7
         Position: 36

In [41]: # to extract maximum/largest numeric value from a string.
         string = "My marks in each semester are: 947, 896, 926, 524, 734, 950, 642"
         pattern = re.findall(r'\d+', string)
         pattern = [int(value) for value in pattern]
         max_value = max(pattern)
         print(max_value)

         950

In [15]: # to insert spaces between words starting with capital letters.
         string = "RegularExpressionIsAnImportantTopicInPython"
         x = re.findall('[A-Z][a-z]*', string)
         print(' '.join((x)))

         Regular Expression Is An Important Topic In Python
```

```
In [57]: # to find sequences of one upper case letter followed by lower case letters.
         string = "fhyugyu Dgdg DGkjghksj Fsdklkjls kjsh 4hjkjdhgyg"
         pattern = r'[A-Z][a-z]+'
         result = re.findall(pattern, string)
         print(result)
```

```
['Dgdg', 'Gkjghksj', 'Fsdklkjls']
```

```
In [8]: # to remove continuous duplicate words from Sentence using Regular Expression.
        string = "Hello hello world world"
        regex = r'\b(\w+)(?:\b\W+\1\b)+'
        x = re.sub(regex, r'\1', string)
        print(x)
```

```
Hello hello world
```

```
In [21]: # to accept string ending with alphanumeric character.
         string = "dgger sfeg zdgdr"
         pattern = r'\w$'
         match = re.search(pattern, string)
         print(match)
```

```
<regex.Match object; span=(15, 16), match='r'>
```

```
In [13]: # to extract the hashtags.
         string = """RT @kapil_kausik: #Doltiwal I mean #xyzabc is "hurt" by #Demonetization as the same has rendered USELESS <ed><U+00A0>
         x = re.findall('#[\w+][a-zA-Z0-9]+', string)
         print(x)

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

```
In [55]: # to remove <U+..> like symbols
         string = "@Jags123456 Bharat band on 28??<ed><U+00A0><U+00BD><ed><U+00B8><U+0082>Those who  are protesting #demonetization  are a
         pattern = r"<U\+\w{4}>"
         x = re.sub(pattern, "", string)
         print(x)

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

```
In [56]: # to extract dates from the text stored in the text file.
         with open('Ron.txt', 'r') as file:
             text = file.read()
             pattern = r'\d{2}-\d{2}-\d{4}'
             dates = re.findall(pattern, text)
         for date in dates:
             print(date)

         12-09-1992
         15-12-1999
```

```
In [44]: # to remove all words from a string of length between 2 and 4.
         string = "The following example creates an ArrayList with a capacity of 50 elements. 4 elements are then added to the ArrayList a
         pattern = re.compile(r'\b\w{2,4}\b')
         x = re.sub(pattern, '', string)
         print(x)
```

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