disc05-extra

July 2, 2020

1 Extra Practice with Regular Expressions

Collaboration Policy

Data science is a collaborative activity. While you may talk with others about the homework, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** at the top of your solution.

1.0.1 This assignment is optional and will not be graded.

1.1 Collaborators

Write names in this cell:

[1]:
```
import pandas as pd
import numpy as np
import re
```

1.2 Objectives:

You will practice the basic usage of regular expressions and also learn to use `re` module in Python. Some of the materials are based on the tutorial at http://opim.wharton.upenn.edu/~sok/idtresources/python/regex.pdf. As you work through this assignment, we suggest you to use the website http://regex101.com, especially when you have difficulties matching your answer with the asked part of string.

2 Question 1

In this question, write patterns that match the given sequences. It may be as simple as the common letters on each line.

2.1 Question 1a

Write a single regular expression to match the following strings without using the | operator.

1. **Match:** `abcdefg`
2. **Match:** `abcde`
3. **Match:** `abc`
4. **Skip:** `c abc`

```
BEGIN QUESTION
name: q1a
```

[2]:
```
regx1 = r"" # fill in your pattern
# BEGIN SOLUTION
regx1 = r"^abc[\w]*"
# END SOLUTION
```

[3]:
```
# TEST
"|" not in regx1
```

[3]:
```
True
```

[4]:
```
# TEST
re.search(regx1, "abc").group()
```

[4]:
```
'abc'
```

[5]:
```
# TEST
re.search(regx1, "abcde").group()
```

[5]:
```
'abcde'
```

[6]:
```
# TEST
re.search(regx1, "abcdefg").group()
```

[6]:
```
'abcdefg'
```

[7]:
```
# TEST
re.search(regx1, "c abc") is None
```

[7]:
```
True
```

2.2 Question 1b

Write a single regular expression to match the following strings without using the | operator.

1. **Match:** `can`

2. **Match:** `man`
3. **Match:** `fan`
4. **Skip:** `dan`
5. **Skip:** `ran`
6. **Skip:** `pan`

```
BEGIN QUESTION
name: q1b
```

[8]:
```
regx2 = r"" # fill in your pattern
# BEGIN SOLUTION
regx2 = r"^([cmf]an)"
# END SOLUTION
```

[9]:
```
# TEST
"|" not in regx2
```

[9]:
```
True
```

[10]:
```
# TEST
re.match(regx2, 'can').group()
```

[10]:
```
'can'
```

[11]:
```
# TEST
re.match(regx2, 'fan').group()
```

[11]:
```
'fan'
```

[12]:
```
# TEST
re.match(regx2, 'man').group()
```

[12]:
```
'man'
```

[13]:
```
# TEST
re.match(regx2, 'dan') is None
```

[13]:
```
True
```

[14]:
```
# TEST
re.match(regx2, 'ran') is None
```

[14]:
```
True
```

[15]:
```
# TEST
re.match(regx2, 'pan') is None
```

[15]:
```
True
```

3 Question 2

Now that we have written a few regular expressions, we are now ready to move beyond matching. In this question, we'll take a look at some methods from the `re` package.

3.1 Question 2a:

Write a Python program to extract and print the numbers of a given string.

1. **Hint:** use `re.findall`
2. **Hint:** use `\d` for digits and one of either `*` or `+`.

```
BEGIN QUESTION
name: q2a
```

[16]:
```
text_q2a = "Ten 10, Twenty 20, Thirty 30"

res_q2a = ...
# BEGIN SOLUTION
res_q2a = re.findall(r"\d+", text_q2a)
# END SOLUTION

res_q2a
```

[16]:
```
['10', '20', '30']
```

[17]:
```
# TEST
res_q2a
```

[17]:
```
['10', '20', '30']
```

3.2 Question 2b:

Write a Python program to replace at most 2 occurrences of space, comma, or dot with a colon.

Hint: use `re.sub(regex, "newtext", string, number_of_occurences)`

```
BEGIN QUESTION
name: q2b
```

[18]:
```
text_q2b = 'Python Exercises, PHP exercises.'
res_q2b = ... # Hint: use re.sub()
# BEGIN SOLUTION
res_q2b = re.sub("[ ,.]", ":", text_q2b, 2)
# END SOLUTION
```

```
res_q2b
```

[18]:
```
'Python:Exercises: PHP exercises.'
```

[19]:
```
# TEST
res_q2b
```

[19]:
```
'Python:Exercises: PHP exercises.'
```

3.3 Question 2c:

Write a Python program to extract values between quotation marks of a string.

Hint: use `re.findall`

```
BEGIN QUESTION
name: q2c
```

[20]:
```
text_q2c = '"Python", "PHP", "Java"'
res_q2c = ...
# BEGIN SOLUTION
res_q2c = re.findall(r'"(.*?)"', text_q2c)
# END SOLUTION

res_q2c
```

[20]:
```
['Python', 'PHP', 'Java']
```

[21]:
```
# TEST
res_q2c
```

[21]:
```
['Python', 'PHP', 'Java']
```

3.4 Question 2d:

Write a regular expression to extract and print the quantity and type of objects in a string. You may assume that a space separates quantity and type, ie. `"{quantity} {type}"`. See the example string below for more detail.

1. **Hint:** use `re.findall`
2. **Hint:** use `\d` for digits and one of either `*` or `+`.

```
BEGIN QUESTION
name: q2d
```

```
[22]: text_q2d = "I've got 10 eggs that I stole from 20 gooses belonging to 30 giants.
       ↪"

      res_q2d = ...
      # BEGIN SOLUTION
      res_q2d = re.findall(r"\d+\s[^\s.]+", text_q2d)
      # END SOLUTION

      res_q2d
```

```
[22]: ['10 eggs', '20 gooses', '30 giants']
```

```
[23]: # TEST
      res_q2d
```

```
[23]: ['10 eggs', '20 gooses', '30 giants']
```

3.5 Question 2e:

Write a regular expression to replace all vowels with a lowercase letter "o". Given that address is a string, use re.sub to change "123 Orange Street" into "123 orongo Stroot".

Hint: use `re.sub(regex, "newtext", string, number_of_occurences)`

```
BEGIN QUESTION
name: q2e
```

```
[24]: text_q2e = "123 Orange Street"

      res_q2e = ...
      # BEGIN SOLUTION
      res_q2e = re.sub(r"[aeiuAEIOU]", "o", text_q2e)
      # END SOLUTION

      res_q2e
```

```
[24]: '123 orongo Stroot'
```

```
[25]: # TEST
      res_q2e
```

```
[25]: '123 orongo Stroot'
```

3.6 Question 2f:

This question comes from the RegEx puzzle from lecture. Fill in the regular expression in the variable pattern below so that after it executes, day is 26, month is Jan, and year is 2014.

1. **Hint:** use `re.findall`
2. **Hint:** pay attention to the data type after using `re.findall`
3. **Hint:** use \[and \/ to match the character '[' and '/'.

```
BEGIN QUESTION
name: q2f
```

[26]:
```
text_q2f = '169.237.46.168 - - [26/Jan/2014:10:47:58 -0800] \
"GET /stat141/Winter04/ HTTP/1.1" 200 2585 \
"http://anson.ucdavis.edu/courses/"'
pattern = ... # Hint: only pattern has to be regular expression
day = ... # day, month, year all depend on pattern
month = ...
year = ...
# BEGIN SOLUTION
pattern = re.findall(r"\[(.+)\/(.+)\/([^:]+).*\]", text_q2f)[0]
# END SOLUTION
pattern
```

[26]:
```
('26', 'Jan', '2014')
```

[27]:
```
# TEST
pattern
```

[27]:
```
('26', 'Jan', '2014')
```

[28]:
```
# TEST
pattern[0] == '26'
```

[28]:
```
True
```

[29]:
```
# TEST
pattern[1] == 'Jan'
```

[29]:
```
True
```

[30]:
```
# TEST
pattern[2] == '2014'
```

[30]:
```
True
```

Congrats! You have finished this assignment.