MIT World Peace University

Python Programming
Second Year B. Tech, Semester 4

Learning Basics of *regex*

Assignment No. 9

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

April 30, 2023

Contents

1 Aim

To find a string or set of strings using special sequence of characters that uses a search pattern.

2 Problem Statement

Create a Regular Expression and implement the following

1. Recognize the following strings: 'bat,' 'bit,' 'but,' 'hat,' 'hit,' or 'hut.'
2. Match any pair of words separated by a single space, i.e., first and last names.
3. Match any word and single letter separated by a comma and single space, as in last name, first initial.

3 Objectives

1. To learn and implement Functions of Regular Expression.

4 Theory

4.1 Metacharacters and Special Sequences

4.1.1 Metacharacters

Metacharacters are characters with a special meaning:

1. **^** - Starts with
2. **$** - Ends with
3. **.** - Any character
4. **|** - Either or
5. **[]** - Range
6. **()** - Capture and group
7. \ - Escape character
8. \d - Digits
9. \D - Not a digit
10. \w - Word character
11. \W - Not a word character
12. \s - Whitespace
13. \S - Not a whitespace
14. \b - Word boundary
15. \B - Not a word boundary

16. **A** - Beginning of string
17. **Z** - End of string
18. **z** - End of string
19. **G** - End of previous match
20. **n** - New line
21. **t** - Tab
22. \ - Backslash
23. **(escape)** - Escape character
24. **{number}** - Occurrences
25. **{number, number}** - Range of occurrences
26. **{number, }** - Minimum occurrences
27. * - Zero or more occurrences
28. **+** - One or more occurrences
29. **?** - Zero or one occurrences

They are used to define the search pattern. They are used in the search() methods to search a string for a match.

4.2 Regular Expression Functions

4.2.1 re.findall()

The re.findall() method returns a list of strings containing all matches. If there are no matches, an empty list is returned.

Parameters: This method takes following arguments :

- pattern: A regular expression to match the string.
- string: A string to be searched.
- flags: We can specify different flags using bitwise OR (|). These are modifiers, which are listed in the table below.

```
    pattern = r"eggs"
    string = "bacon and eggs"
    re.findall(pattern, string, flags=0)
    # output = ['eggs']
```

4.2.2 re.split()

The re.split method splits the string where there is a match and returns a list of strings where the splits have occurred.

Parameters: This method takes following arguments :

- pattern: A regular expression to match the string.

- string: A string to be split on basis of pattern matching.
- maxsplit: It is a number, which tells us to split the string into maximum of provided number of times. If not provided, the default is -1 that is, all occurrences.
- flags: It is an optional parameter that can be used to control different regular expression operations.

```
    pattern = r"\s"
    string = "bacon and eggs"
    re.split(pattern, string, maxsplit=0, flags=0)
    # output = ['bacon', 'and', 'eggs']

    re.split(pattern, string, maxsplit=1, flags=0)

    # output = ['bacon', 'and eggs']
```

4.2.3 re.sub()

The re.sub() method replaces all occurrences of the RE pattern in string with repl, substituting all occurrences unless max provided. This method returns modified string.

Parameters: This method takes following arguments :

- pattern: A regular expression to match the string.
- repl: A substitute of the replace string.
- string: A string to be matched.
- count: Maximum number of occurrences that need to be replaced.
- flags: It is an optional parameter that can be used to control different regular expression operations.

```
    pattern = r"eggs"
    string = "bacon and eggs"
    repl = "spam"
    re.sub(pattern, repl, string, count=0, flags=0)
    # output = 'bacon and spam'
```

4.2.4 re.search()

The re.search() method takes two arguments: a pattern and a string. The method looks for the first location where the RegEx pattern produces a match with the string.

If the search is successful, re.search() returns a match object; if not, it returns None.

Parameters: This method takes following arguments :

- pattern: A regular expression to match the string.
- string: A string to be searched.
- flags: We can specify different flags using bitwise OR (`|`). These are modifiers, which are listed in the table below.

```
    pattern = r"eggs"
    string = "bacon and eggs"
    re.search(pattern, string, flags=0)

    # output = <re.Match object; span=(10, 14), match='eggs'>

    pattern = r"bacon"

    re.search(pattern, string, flags=0)
    # output = <re.Match object; span=(0, 5), match='bacon'>

    pattern = r"sausage"
    re.search(pattern, string, flags=0)

    # output = None
```

4.2.5 re.match()

The re.match() function returns a match object on success, None on failure. We use group(num) or groups() function of match object to get matched expression.

Parameters: This method takes a regular expression pattern and a string and searches for that pattern with the string.

```
    pattern = r"eggs"
    string = "bacon and eggs"
    re.match(pattern, string, flags=0)
    # output = None

    pattern = r"bacon"
    re.match(pattern, string, flags=0)
    # output = <re.Match object; span=(0, 5), match='bacon'>

    pattern = r"sausage"

    re.match(pattern, string, flags=0)
    # output = None
```

5 Input and Output

5.1 Input

Some string that has a pattern that needs to be matched is taken as input.

5.2 Output

The output is the string that matches the pattern.

6 Requirements

1. Python 3.7 or above

7 Code

```
[2]: import re
```

```
[3]: text = 'The rain in Spain'
     x = re.findall('ai', text)
```

```
[4]: x
```

```
[4]: ['ai', 'ai']
```

7.0.1 Search

```
[5]: re.search('ra', text)
```

```
[5]: <re.Match object; span=(4, 6), match='ra'>
```

```
[6]: re.split('\s', text)
```

```
[6]: ['The', 'rain', 'in', 'Spain']
```

```
[7]: re.sub('\s', '-', text)
```

```
[7]: 'The-rain-in-Spain'
```

```
[8]: text
```

```
[8]: 'The rain in Spain'
```

7.0.2 Assignment stuff

1. Recognize the following strings “but, bit, bat, hit, hut, hit”

```
[10]: text = "Ram is playing with his cat, who is sitting near his cricket bat. The␣
       ↪cat is fat. It doesnt like chit chat. It only likes to take a nap. Ram wears a␣
       ↪hat, but the every time he wears the hat, Ram's cat would hit him. If they␣
       ↪fight a lot, then Ram just leaves his hut and finds another cat. "
```

```
[11]: re.findall(r'\b[bh][aui]t\b', text)
```

```
[11]: ['bat', 'hat', 'but', 'hat', 'hit', 'hut']
```

7.0.3 2. Match any pair of words separated by a single space. That is first and last names

```
[12]: text = "There are name names in this class, like Parth Zarekar, Nishad Wanjari"
```

```
[16]: re.findall(r'\s', text)
```

```
[16]: [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

7.0.4 3. Match any word and single letter separated by a comma and a single space.

```
[17]: text = "Parth, Zarekar and Nishad, Wanjari are going to talk to Vedeng, Khare."
```

```
[21]: re.findall(r'\b\w+,\s.', text)
```

```
[21]: ['Parth, Z', 'Nishad, W', 'Vedeng, K']
```

```
[ ]:
```

8 Conclusion

Studied python Regular Expression functions for string handling.

9 FAQ

1. **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'.**

```python
import re
text = 'The quick brown fox jumps over the lazy dog.'
print(re.findall(r'fox|dog|horse', text))
# Output: ['fox', 'dog']
```

2. **Explain the findall () and sub () methods?** The re.findall() method returns a list of strings containing all matches. The Parameters used in this function are:

   - pattern which is the regular expression to be matched.
   - string which is the string where pattern would be searched to fetch the matching strings.
   - flags which are optional and can be used to modify the meaning of the given pattern.

   The re.sub() method returns the string obtained after replacing or substituting the specified patterns in the string. The Parameters used in this function are:

   - pattern which is the regular expression to be matched.
   - repl which is the substitute for the matched string.
   - string which is the string where pattern would be searched to fetch the matching strings.
   - count which is optional and can be used to specify the number of occurrences to be replaced.

   Here's an example of findall() and sub() methods:

```python
import re
text = 'The quick brown fox jumps over the lazy dog.'
print(re.findall(r'fox|dog|horse', text))
# Output: ['fox', 'dog']
print(re.sub(r'fox|dog', 'cat', text))
# Output: The quick brown cat jumps over the lazy cat.
```