Python Regex sub()

Summary: in this tutorial, you'll learn about the Python regex `sub()` function that returns a string after replacing the matched pattern in a string with a replacement.

Introduction to the Python regex sub function

The `sub()` is a function in the built-in `re` module that handles regular expressions (https://www.pythontutorial.net/python-regex/python-regular-expressions/). The `sub()` function has the following syntax:

```
re.sub(pattern, repl, string, count=0, flags=0)
```

In this syntax:

- `pattern` is a regular expression that you want to match. Besides a regular expression, the `pattern` can be `Pattern` object.
- `repl` is the replacement
- `string` is the input string

- `count` parameter specifies the maximum number of matches that the `sub()` function should replace. If you pass zero to the `count` parameter or completely skip it, the `sub()` function will replace all the matches.
- `flags` is one or more regex flags (https://www.pythontutorial.net/python-regex/python-regex-flags/) that modify the standard behavior of the pattern.

The `sub()` function searches for the pattern in the string and replaces the matched strings with the replacement (`repl`).

If the `sub()` function couldn't find a match, it returns the original string. Otherwise, the `sub()` function returns the string after replacing the matches.

Note that the `sub()` function replaces the leftmost non-overlapping occurrences of the pattern. And you'll see it in detail in the following example.

Python regex sub function examples

Let's take some examples of using the regex `sub()` function.

1) Using the regex sub() function to return the plain phone number

The following example uses the `sub()` function to turn the phone number `(212)-456-7890` into `2124567890` :

```
import re

phone_no = '(212)-456-7890'
pattern = '\D'
result = re.sub(pattern, '',phone_no)

print(result)
```

Output:

```
2124567890
```

In this example, the `\D` is an inverse digit character set (https://www.pythontutorial.net/python-regex/python-regex-character-set/) that matches any single character which is not a digit. Therefore, the `sub()` function replaces all non-digit characters with the empty string `''`.

2) Using the regex sub() function to replace the leftmost non-overlapping occurrences of a pattern

The following example replaces the `00` with the `''` in the string `'000000'`:

```
import re

pattern = '00'
s = '00000'
result = re.sub(pattern,'',s)

print(result)
```

Output:

```
0
```

In this example, we replace two zeros with empty strings. So the first two are matched and replaced, then the following two zeroes are matches and replaced too, and finally, the last digit remains unchanged.

3) Using the regex sub() with a backreference example

The following example uses the `sub()` function to replace the text surrounded with (`*`) (it's markdown format by the way) with the `<b>` tag in HTML:

```
import re

s = 'Make the World a *Better Place*'
```

```
pattern = r'\*(.*?)\*'
replacement = r'<b>\1<\\b>'
html = re.sub(pattern, replacement, s)

print(html)
```

Output:

```
import re

s = 'Make the World a *Better Place*'
pattern = r'\*(.*?)\*'
replacement = r'<b>\1<\\b>'
html = re.sub(pattern, replacement, s)

print(html)
```

Output:

```
Make the World a <b>Better Place<\b>
```

In this example, the pattern `r'\*(.*?)\*'` find the text that begins and ends with the asterisk (`*`). It has a capturing group that captures the text between asterisks (`*`).

The replacement is a regular expression with a backreference (https://www.pythontutorial.net/python-regex/python-regex-backreferences/) . The backreference `\1` refers to the first group in the pattern, which is the text between the asterisks (`*`).

4) Using the regex sub() function with the replacement as a function

Suppose you have a list of strings where each element contain both alphabet and number:

```
l = ['A1','A2','A3']
```

And you want to square the number in each list element. For example, A1 becomes A1, A2 becomes A4, and A3 becomes A9. To do this, you can use the `sub()` function.

The second argument of the `sub()` function (`repl`) can be a function. In this case, the `sub()` function will call this function for every non-overlapping occurrence of the pattern.

This function (`repl`) takes a single `Match` object argument and returns the replacement string.

The following illustrates how to use the second argument as a function:

```
import re

def square(match):
    num = int(match.group())
    return str(num*num)

l = ['A1','A2','A3']
pattern = r'\d+'
new_l = [re.sub(pattern, square, s) for s in l]

print(new_l)
```

Output:

```
['A1', 'A4', 'A9']
```

How it works.

First, define a list of strings:

```
l = ['A1','A2','A3']
```

Second, define a pattern `\d+` that match one or more digits:

```
pattern = r'\d+'
```

Third, replace the digits with their squares by calling the `sub()` function and passing the `square()` function:

```
new_l = [re.sub(pattern, square, s) for s in l]
```

Finally, define the `square()` function that squares the matched digit and returns it:

```
def square(match):
    num = int(match.group())
    return str(num*num)
```

Summary

- Use the Python regex `sub()` function to replace the occurrences of matches of a pattern with a replacement.