Python Regex split()

Summary: in this tutorial, you'll learn how to use the Python regex `split()` function to split a string at the occurrences of matches of a regular expression.

Introduction to the Python regex split() function

The built-in `re` module provides you with the `split()` function that splits a string by the matches of a regular expression (https://www.pythontutorial.net/python-regex/python-regular-expressions/) .

The `split()` function has the following syntax:

```
split(pattern, string, maxsplit=0, flags=0)
```

In this syntax:

- `pattern` is a regular expression whose matches will be used as separators for splitting.
- `string` is an input string to split.
- `maxsplit` determines at most the splits occur. Generally, if the `maxsplit` is one, the resulting list will have two elements. If the `maxsplit` is two, the resulting list will have three elements, and so on.

- `flags` parameter is optional and defaults to zero. The `flags` parameter accepts one or more regex flags (https://www.pythontutorial.net/python-regex/python-regex-flags/) . The `flags` parameter changes how the regex engine matches the pattern.

The `split()` function returns a list of substrings split by the matches of the pattern in the string.

If the `pattern` contains one or more capturing groups (https://www.pythontutorial.net/python-regex/python-regex-capturing-group/) , the `split()` function will return the text of all groups as elements of the resulting list.

If the `pattern` contains a capturing group that matches the start of a string, the `split()` function will return a resulting list with the first element being as an empty string. This logic is the same for the end of the string.

Python regex split() function examples

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

1) Using the split() function to split words in a sentence

The following example uses the `split()` function to split the words in a sentence:

```
import re

s = 'A! B. C D'
pattern = r'\W+'

l = re.split(pattern, s)
print(l)
```

In this example, the `\W+` is the inverse of the word character set (https://www.pythontutorial.net/python-regex/python-regex-character-set/) that matches one or more characters that are not the word characters.

Output:

```
['A', 'B', 'C', 'D']
```

2) Using the split() function with the maxsplit argument

The following example uses the `split()` function that splits a string with two splits at non-word characters:

```
import re

s = 'A! B. C D'
pattern = r'\W+'

l = re.split(pattern, s, 2)
print(l)
```

Output:

```
['A', 'B', 'C D']
```

Because we split the string with two splits, the resulting list contains three elements. Notice that the `split()` function returns the remainder of a string as the final element in the resulting list.

3) Using the split() function with a capturing group

The following example uses the `split()` function that splits a string with the `\W+` pattern that contains a capturing group:

```
import re

s = 'A! B. C D'
pattern = r'(\W+)'

l = re.split(pattern, s, 2)
print(l)
```

Output:

```
['A', '! ', 'B', '. ', 'C D']
```

In this example, the `split()` function also returns the text of the group in the resulting list.

4) Using the split() function

The following example uses the `split()` function where the separator contains a capturing group that matches the start of the string:

```
import re

s = '...A! B. C D'
pattern = r'\W+'

l = re.split(pattern, s)
print(l)
```

In this case, the `split()` function returns a list with the first element is an empty string:

```
['', 'A', 'B', 'C', 'D']
```

Similarly, if the separator contains the capturing groups and it matches the end of the string, the resulting list will have the last element as an empty string:

```
import re

s = 'A! B. C D...'
pattern = r'\W+'

l = re.split(pattern, s)
print(l)
```

Output:

```
['A', 'B', 'C', 'D', '']
```

Summary

- Use the Python regex `split()` function to split a string using sepators as the matches of a regular expression.