Python Regex Alternation

Summary: in this tutorial, you'll learn about Python regex alternation, which behaves like the "OR" operator in regular expressions.

Introduction to the Python regex alternation

To represent an alternation in regular expressions (https://www.pythontutorial.net/python-regex/python-regular-expressions/), you use the pipe operator (`|`). The pipe operator is called the **alternation**. It is like the `or` operator in Python.

The following regular expression uses an alternation to match either the literal string `complex` and `simple`:

```
'simple|complex'
```

For example, the following program uses the above regular expression to match either the literal string `simple` or `complex`:

```
import re
```

```
s = 'simple is better than complex'
pattern = r'simple|complex'

matches = re.findall(pattern,s)
print(matches)
```

Output:

```
['simple', 'complex']
```

Python regex alternation examples

Let's take more examples of using the regex alternation.

1) Use Python regex alternation for matching time in hh:mm format

To match a time string in the `hh:mm` format, you can combine the `\d` character set (https://www.pythontutorial.net/python-regex/python-regex-character-set/) with the quantifiers (https://www.pythontutorial.net/python-regex/python-regex-quantifiers/) `{}` :

```
'\d{2}:\d{2}'
```

In this pattern:

- `\d{2}` matches two digits.
- `:` matches the colon character.
- `\d{2}` matches two digits.

However, the rule `\d{2}` also matches a number that is not a valid hour or minute, such as `99` .

To fix this, you can use the regex alternation.

If the valid hour ranges from `01` to `23` , you can use the following pattern to match the hour part:

```
[01]\d|2[0-3]
```

In this pattern:

- `[01]` matches a single digit 0 or 1
- `\d` matches a single digit from 0 to 9
- `[01]\d` matches 00, 01 to 19
- `2` matches the digit 2
- `[0-3]` matches a single digit from 0 to 3 including 0, 1, 2, 3
- `2[0-3]` matches two digits 20, 21, 22, and 23.

Therefore, the `[01]\d|2[0-3]` matches two digits from 00 to 23

Because the valid minute ranges from `00` to `59`, you can use the following pattern to match it:

```
[0-5]\d
```

The following regular expression combines the two rules above to match the time in the `hh:mm` format:

```
'[01]\d|2[0-3]:[0-5]\d'
```

However, this regular expression will not work as expected. For example:

```
import re

s = '09:30 30:61 22:30 25:99'
pattern = r'[01]\d|2[0-3]:[0-5]\d'

matches = re.finditer(pattern, s)
for match in matches:
    print(match.group())
```

Output:

```
09
22:30
```

In this example, the regex engine treats pattern `[01]\d|2[0-3]:[0-5]\d` as two main parts separated by the alternation:

```
[01]\d
OR
2[0-3]):([0-5]\d)
```

To fix it, you need to wrap the alternation inside parentheses to indicate that only that part is alternated, not the whole expression like this:

```
([01]\d|2[0-3]):[0-5]\d
```

Now, the program works as expected:

```
import re

s = '09:30 30:61 22:30 25:99'
pattern = r'([01]\d|2[0-3]):[0-5]\d'

matches = re.finditer(pattern, s)
for match in matches:
    print(match.group())
```

Output:

```
09:30
22:30
```

Summary

- The regex alternation `x | y` matches either `x` or `y`.
- The regex alternation is like an OR operator in regular expressions.
- Place the alternation part inside parentheses `()` to express that only that part is alternated.