Python Regex Greedy

Summary: in this tutorial, you'll learn about the Python regex greedy mode and how to change the mode from greedy to non-greedy.

By default, all quantifiers (https://www.pythontutorial.net/python-regex/python-regex-quantifiers/) work in a greedy mode. It means that the quantifiers will try to match their preceding elements as much as possible.

Let's start with an example to understand how the regex greedy mode works.

The unexpected result with greedy mode

Suppose you have the following HTML fragment that represents a button element:

```
s = '<button type="submit" class="btn">Send</button>'
```

And you want to match the texts within the quotes (`""`) like `submit` and `btn` .

To do that, you may come up with the following pattern that includes the quote ("), the dot (`.`) character set (https://www.pythontutorial.net/python-regex/python-regex-character-set/) and the (`+`) quantifier (https://www.pythontutorial.net/python-regex/python-regex-quantifiers/) :

```
".+"
```

The meaning of the pattern is as follows:

- `"` starts with a quote
- `.` matches any character except the newline
- `+` matches the preceding character one or more times
- `"` ends with a quote

The following uses the `finditer()` function to match the string `s` with the pattern:

```
import re

s = '<button type="submit" class="btn">Send</button>'

pattern = '".+"'
matches = re.finditer(pattern, s)

for match in matches:
    print(match.group())
```

The program displays the following result:

```
"submit" class="btn"
```

The result is not what you expected.

By default, the quantifier (+) runs in the greedy mode, in which it tries to match the preceding element (`".`) as much as possible.

How Python regex greedy mode works

First, the regex engine starts matching from the first character in the string `s` .

Next, because the first character is `<` which does not match the quote (`"`), the regex engine continues to match the next characters until it reaches the first quote (`"`):

Then, the regex engine examines the pattern and matches the string with the next rule `.+` .

Because the `.+` rule matches a character one or more times, the regex engine matches all characters until it reaches the end of the string:

After that, the regex engine examines the last rule in the pattern, which is a quote (“). However, it already reaches the end of the string. There’s no more character to match. It is too greedy to go too far.

Finally, the regex engine goes back from the end of the string to find the quote (“). This step is called **backtracking**.

As a result, the match is the following substring which is not what we expected:

```
"submit" class="btn"
```

To fix this issue, you need to instruct the quantifier (`+`) to use the non-greedy (or lazy) mode instead of the greedy mode.

To do that, you add a question mark (`?`) after the quantifier like this:

```
".+?"
```

The following program returns the expected result:

```
import re

s = '<button type="submit" class="btn">Send</button>'
```

```
pattern = '".+?"'
matches = re.finditer(pattern, s)

for match in matches:
    print(match.group())
```

Output:

```
"submit"
"btn"
```

Summary

- By default, all quantifiers use the greedy mode.
- Greedy quantifiers will match their preceding elements as much as possible.