Python Regex fullmatch()

Summary: in this tutorial, you'll learn how to use the Python regex `fullmatch()` to match the whole string with a regular expression.

Introduction to the Python regex fullmatch function

The `fullmatch()` function returns a `Match` object if the whole string matches the search pattern of a regular expression (https://www.pythontutorial.net/python-regex/python-regular-expressions/), or `None` otherwise.

The syntax of the `fullmatch()` function is as follows:

```
re.fullmatch(pattern, string, flags=0)
```

In this syntax:

- `pattern` specifies a regular expression to match.
- `string` specifies the input string.
- `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.

Python regex fullmatch function example

The following example uses the `fullmatch()` function to validate an email address:

```
import re

email = 'no-reply@pythontutorial.net'
pattern = r'[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}'
match = re.fullmatch(pattern, email)

if match is not None:
    print(f'The email "{match.group()}" is valid')
else:
    print(f'The email "{email}"" is not valid')
```

Output:

```
The email "no-reply@pythontutorial.net" is valid
```

The following defines a function that uses the `fullmatch()` function to validate an email address. It returns `True` if the email is valid or raises a `ValueError` exception otherwise:

```
import re

def is_email(s: str) -> bool:
    pattern = r'[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}'
    if re.fullmatch(pattern, s) is None:
        raise ValueError(f'The {s} is not a valid email address')

    return True
```

And you can use the `is_email()` function to validate an email like this:

```
if __name__ == '__main__':
    try:
       if is_email('no-reply@pythontutorial'):
           print('The email is valid')
    except ValueError as e:
        print(e)
```

Output:

```
The no-reply@pythontutorial is not a valid email address
```

Python regex fullmatch vs match

Both `fullmatch()` and `match()` functions return a `Match` object if they find a match.

The `fullmatch()` function matches the whole string with a pattern while the `match()` function only finds a match at the beginning of the string. See the following example:

```
import re

s = 'Python 3'
pattern = 'Python'

# fullmatch
match = re.fullmatch(pattern, s)
if match is not None:
    print('fullmatch:', match.group())

# search
match = re.match(pattern, s)
if match is not None:
    print('match:', match.group())
```

Output:

```
match: Python
```

In this example, the `fullmatch()` returns `None` because the pattern `Python` only matches the beginning of the string, not the whole string.

On the other hand, the `match()` function matches the pattern at the beginning of the string and returns the match.

Python fullmatch vs. search

Both `fullmatch()` and `search()` functions return a `Match` object if they find a match of a pattern in a string. However, the `fullmatch()` matches the whole string while the `search()` matches anywhere in the string.

For example:

```
import re

s = 'Python 3'
pattern = '\d'

# fullmatch
match = re.fullmatch(pattern,s)
if match is not None:
    print(match.group())

# search
match = re.search(pattern,s)
if match is not None:
    print(match.group()) # 3
```

Output:

```
3
```

In this example, the pattern `\d` matches a single digit. The `fullmatch()` function returns `None` because the whole string `'Python 3'` doesn't match.

However, the `search()` function returns a match because it could find the digit 3 at the end of the string.

Summary

- Use the Python regex `fullmatch()` function to check if the whole string matches a pattern of a regular expression.