Python Regex Character Set

Summary: in this tutorial, you learn about character sets in regular expressions including digits, words, whitespace, and the dot (.).

Introduction to Python regex character sets

A character set (or a character class) is a set of characters, for example, digits (from 0 to 9), alphabets (from a to z), and whitespace.

A character set allows you to construct regular expressions (https://www.pythontutorial.net/python-regex/python-regular-expressions/) with patterns that match a string with one or more characters in a set.

\d: digit character set

Regular expressions use `\d` to represent a digit character set that matches a single digit from `0` to `9` .

The following example uses the `finditer()` function to match every single digit in a string using the `\d` character set:

```
import re

s = 'Python 3.0 was released in 2008'
matches = re.finditer('\d', s)
for match in matches:
    print(match.group())
```

Output:

```
3
0
2
0
0
8
```

To match a group of two digits, you use the `\d\d`. For example:

```
import re

s = 'Python 3.0 was released in 2008'
matches = re.finditer('\d\d', s)
for match in matches:
    print(match.group())
```

Output:

```
20
08
```

Similarly, you can match a group of four digits using the `\d\d\d\d` pattern:

```
import re

s = 'Python 3.0 was released in 2008'
matches = re.finditer('\d\d\d\d', s)
for match in matches:
    print(match.group())
```

Output:

```
2008
```

Later, you'll learn how to use quantifiers (https://www.pythontutorial.net/python-regex/python-regex-quantifiers/) to shorten the pattern. So instead of using the `\d\d\d\d` pattern, you can use the shorter one like `\d{4}`

\w: the word character set

Regular expressions use `\w` to represent the word character set. The `\w` matches a single ASCII character including Latin alphabet, digit, and underscore (`_`).

The following example uses the `finditer()` function to match every single word character in a string using the `\w` character set:

```
import re

s = 'Python 3.0'
matches = re.finditer('\w', s)
for match in matches:
    print(match.group())
```

Output:

```
P
y
t
h
o
n
3
0
```

Notice that the whitespace and `.` are not included in the matches.

\s : whitespace character set

The `\s` matches whitespace including a space, a tab, a newline, a carriage return, and a vertical tab.

The following example uses the whitespace character set to match a space in a string:

```
import re

s = 'Python 3.0'
matches = re.finditer('\s', s)
for match in matches:
    print(match)
```

Output:

```
<re.Match object; span=(6, 7), match=' '>
```

Inverse character sets

A character set has an inverse character set that uses the same letter but in uppercase. The following table shows the character sets and their inverse ones:

Character set	Inverse character set	Description
\d	\D	Match a single character except for a digit
\w	\W	Match a single character that is not a word character
\s	\S	Match a single character except for whitespace

The following example uses the `\D` to match the non-digit from a phone number:

```
import re

phone_no = '+1-(650)-513-0514'
matches = re.finditer('\D', phone_no)
for match in matches:
    print(match.group())
```

Output:

```
+
-
(
)
-
-
```

To turn the phone number +1-(650)-513-0514 into the 16505130514, you can use the sub() function:

```
import re

phone_no = re.sub('\D', '', '+1-(650)-513-0514')
print(phone_no)
```

Output:

```
16505130514
```

In this example, the `sub()` function replaces the character that matches the pattern `\D` with the literal string `''` in the formatted phone number.

The dot(.) character set

The dot (`.`) character set matches any single character except the new line (`\n`). The following example uses the dot (.) character set to match every single character but the new line:

```
import re

version = "Python\n4"
matches = re.finditer('.', version)
for match in matches:
    print(match.group())
```

Output:

```
P
y
t
h
o
n
4
```

Summary

- Use `\d` character set to match any single digit.
- Use `\w` character set to match any single word character.

- Use `\s` character set to match any whitespace.
- The `\D` , `\W` , `\S` character set are the inverse sets of `\d` , `\w` , and `\s` character set.
- Use the dot character set (`.`) to match any character but a new line.