REGEX(regular expression)

Regular Expressions in Python

Regular expressions, or "regex" for short, are a powerful tool for working with strings and text data in Python. They allow you to match and manipulate strings based on patterns, making it easy to perform complex string operations with just a few lines of code.

Metacharacters in regular expressions

```
[]   Represent a character class
^    Matches the beginning
$    Matches the end
.    Matches any character except newline
?    Matches zero or one occurrence.
|    Means OR (Matches with any of the characters
     separated by it.
*    Any number of occurrences (including 0 occurrences)
+    One or more occurrences
{}   Indicate number of occurrences of a preceding RE
     to match.
()   Enclose a group of REs
```

Find list of more meta characters here - https://www.ibm.com/docs/en/rational-clearquest/9.0.1?topic=tags-meta-characters-in-regular-expressions

Importing re Package

In Python, regular expressions are supported by the `re` module. The basic syntax for working with regular expressions in Python is as follows:

```
import re
```

Searching for a pattern in re using re.search() Method

re.search() method either returns None (if the pattern doesn’t
match), or a re.MatchObject that contains information about the matching
part of the string. This method stops after the first match, so this is
best suited for testing a regular expression more than extracting data.
We can use re.search method like this to search for a pattern in regular
expression:

```
# Define a regular expression pattern
pattern = r"expression"
```

```
# Match the pattern against a string
text = "Hello, world!"
```

```
match = re.search(pattern, text)
```

```
if match:
    print("Match found!")
else:
    print("Match not found.")
```

Searching for a pattern in re using re.findall() Method

You can also use the `re.findall` function to find all occurrences of the pattern in a string:

```
import re
pattern = r"expression"
text = "The cat is in the hat."
```

```
matches = re.findall(pattern, text)
```

```
print(matches)
# Output: ['cat', 'hat']
```

Replacing a pattern

The following example shows how to replace a pattern in a string:

```
import re
pattern = r"[a-z]+at"
text = "The cat is in the hat."
```

```
matches = re.findall(pattern, text)
```

```
print(matches)
# Output: ['cat', 'hat']
```

```
new_text = re.sub(pattern, "dog", text)
```

```
print(new_text)
# Output: "The dog is in the dog."
```

Extracting information from a string

The following example shows how to extract information from a string using regular expressions:

```
import re
```

```
text = "The email address is example@example.com."
```

```
pattern = r"\w+@\w+\.\w+"
```

```
match = re.search(pattern, text)
```

```
if match:
    email = match.group()
    print(email)
# Output: example@example.com
```

Conclusion

Regular expressions are a powerful tool for working with strings and text data in Python. Whether you're matching patterns, replacing text, or extracting information, regular expressions make it easy to perform complex string operations with just a few lines of code. With a little bit of practice, you'll be able to use regular expressions to solve all sorts of string-related problems in Python.