JavaScript-Regular Expressions

==========

Regular expressions, also known as regex or regexp, are a powerful tool used to match patterns in text. In this summary, we'll cover the basics of regular expressions and provide examples using Python's built-in `re` module.

What is a Regular Expression?

A regular expression is a pattern that describes a set of strings. It's a way to describe the structure of text data, including characters, sequences, and patterns.

Basic Elements of Regular Expressions

- **Character Classes**: `[abc]` matches any of `a`, `b`, or `c`
- **Repetition**: `a*` matches zero or more `a`s
- **Concatenation**: `ab` matches the sequence `a` followed by `b`
- **Alternation**: `(a|b)` matches either `a` or `b`

Common Patterns in Regular Expressions

- `\w`: word character (equivalent to `[a-zA-Z0-9_]`)
- `\d`: digit
- `\s`: whitespace character
- `\D`: non-digit character
- `\S`: non-whitespace character

Example 1: Matching a Phone Number

```
 import re

phone_number = "123-456-7890"
pattern = r"\d{3}-\d{3}-\d{4}"

if re.match(pattern, phone_number):
    print("Phone number matches pattern")
else:
    print("Phone number does not match pattern")
```

In this example, the regular expression `\d{3}-\d{3}-\d{4}` matches any string that contains three digits followed by a hyphen, three more digits, and another hyphen, followed by four digits.

Example 2: Finding All Words Containing "e"

```
 import re

text = "Hello world, this is an example sentence."
pattern = r"\b\w*e\w*\b"

words = re.findall(pattern, text)
print(words)  # Output: ['example', 'sentence']
```

In this example, the regular expression `\b\w*e\w*\b` matches any word that contains the character `e`. The `\b` anchors match word boundaries.

Example 3: Replacing All Occurrences of a Pattern

```
 import re

text = "Hello world, this is an example sentence."
pattern = r"\bexample\b"
replacement = "sample"

new_text = re.sub(pattern, replacement, text)
print(new_text)  # Output: "Hello world, this is a sample sentence."
```

In this example, the `re.sub` function replaces all occurrences of the pattern `\bexample\b` with the string `"sample"`.

Conclusion

Regular expressions are a powerful tool for text processing in Python. With the basics and examples provided above, you're ready to start using regex patterns to match and manipulate text data.

Curated by Brajesh Kumar