Regular expressions

Regular Expression or RegEx: is a special sequence of characters that uses a search pattern to find a string or set of strings.

It can detect the presence or absence of a text by matching it with a particular pattern and also can split a pattern into one or more sub-patterns.

Regex Module in Python

Python has a built-in module named **"re"** that is used for regular expressions in Python. We can import this module by using the import statement.

- **import** re

You can use RegEx in Python after importing re module.

Example:

This Python code uses regular expressions to search for the word "portal" in the given string and then prints the start and end indices of the matched word within the string.

```
import re
 s = 'GeeksforGeeks: A computer science portal for geeks'
 match = re.search(r'portal', s)
print('Start Index:', match.start())  #34
print('End Index:', match.end())      #40
```

Basic Functions

1. **re.match(pattern, string)**: Determines if the regular expression matches at the beginning of the string.

```
 result = re.match(r'\d+', '123abc')
 if result:
    print('Match found:', result.group())
```

```
else:
    print('No match')
```

2**. re.search(pattern, string):** Scans through a string, looking for any location where the regular expression pattern matches.

```
result = re.search(r'\d+', 'abc123')
if result:
    print('Match found:', result.group())
else:
    print('No match')
```

3**. re.findall(pattern, string):** Finds all substrings where the regular expression matches and returns them as a list.

```
result = re.findall(r'\d+', 'abc123def456')
print('Matches found:', result)
```

4**. re.finditer(pattern, string):** Finds all substrings where the regular expression matches and returns them as an iterator of match objects.

```
matches = re.finditer(r'\d+', 'abc123def456')
for match in matches:
    print('Match found:', match.group())
```

5**. re.sub(pattern, repl, string):** Replaces occurrences of the pattern with a replacement string.

```
result = re.sub(r'\d+', '#', 'abc123def456')
print('Result:', result)
```

Regular Expression Syntax

- **`.`**: Matches any character except a newline.

- **`^`**: Matches the start of the string.

- **`$`**: Matches the end of the string.

- **`*`**: Matches 0 or more repetitions of the preceding pattern.

- **`+`**: Matches 1 or more repetitions of the preceding pattern.

- **`?`**: Matches 0 or 1 repetition of the preceding pattern.

- **`{m,n}`**: Matches from `m` to `n` repetitions of the preceding pattern.

- **`[...]`**: Matches any single character within the brackets.

- **`(...)`**: Matches the regular expression inside the parentheses and indicates a group.

Examples:

- Match a phone number:

```
phone_number = '123-456-7890'
if re.match(r'\d{3}-\d{3}-\d{4}', phone_number):
    print('Valid phone number')
```

- Extract email addresses:

```
text = 'Contact us at info@example.com or support@example.org.'
emails = re.findall(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text)
print('Emails found:', emails)
```

Using Raw Strings

When writing regular expressions in Python, it's often useful to use raw string literals by prefixing the string with `r`. This tells Python not to interpret backslashes as escape characters.

conclusion

Regular expressions are a powerful tool for matching patterns in text. The `re` module in Python provides a robust set of tools for working with regular expressions. By understanding the syntax and functions provided, you can effectively use regular expressions to solve many text processing problems.