Regular Expression (regex)

A regex pattern, short for "regular expression pattern," is a sequence of characters that defines a search pattern. It is a powerful tool used for text processing and searching within strings. Regex patterns are primarily used to:

- **Search**: You can use regex patterns to search for specific patterns or sequences of characters within a text or string.
- **Match**: Regex patterns can be used to check if a given string matches a particular pattern or conforms to a specific format.
- **Extract**: You can extract substrings from a larger text based on a matching pattern.
- **Replace**: Regex patterns enable you to find and replace specific patterns or text within a string.

Regex patterns consist of a combination of **regular characters and special metacharacters** that define the pattern.

Some common metacharacters and their meanings include:

. (dot): Matches any single character except a newline.

*: Matches zero or more occurrences of the preceding character or group.

+: Matches one or more occurrences of the preceding character or group.

?: Matches zero or one occurrence of the preceding character or group.

| (pipe): Acts as an OR operator, allowing you to match one of multiple patterns.

[] (square brackets): Defines a character class, allowing you to match any character from the set.

() (parentheses): Groups characters or patterns together, creating subpatterns.

^ (caret): Matches the beginning of a line or string.

$ (dollar sign): Matches the end of a line or string.

String operations using regular expressions (regex) in Python

1. What is the purpose of the `re` module in Python?

2. How can you check if a string matches a specific pattern using `re.match`?

3. Explain the difference between `re.match` and `re.search`.

4. How do you split a string into a list using `re.split`?

5. Write a regex pattern to match valid email addresses.

6. What does the `re.findall` function do, and how is it used?

7. How can you replace all occurrences of a pattern in a string using `re.sub`?

8. Write a regex pattern to match a valid URL.

9. What is a greedy match in regular expressions?

10. How do you make a regular expression case-insensitive?

11. What does the `re.IGNORECASE` flag do in `re.compile`?

12. Write a regex pattern to extract phone numbers in a specific format (e.g., (123) 456-7890).

13. Explain the purpose of character classes (e.g., `[A-Za-z]`) in regular expressions.

14. How can you find the start and end positions of a match using `re.search`?

15. What is the difference between a raw string (e.g., `r'\d'`) and a regular string in regex patterns?

16. Write a regex pattern to match dates in the format "MM/DD/YYYY."

17. How do you use the `re.split` function to split a string at multiple delimiters?

18. Explain the role of backslashes (`\`) in escaping special characters within regex patterns.

19. Write a regex pattern to match valid IPv4 addresses.

20. What is the purpose of the `re.compile` function in Python regex?

21. How do you use the `re.finditer` function, and what does it return?

22. Write a regex pattern to match HTML tags in a string.

23. Explain the difference between the `*` and `+` quantifiers in regex.

24. What does the `re.DOTALL` flag do in `re.compile`, and when is it useful?

25. How can you extract the domain name from a list of email addresses using regex?

26. Write a regex pattern to match valid time durations in the format "HH:MM:SS."

27. What are non-capturing groups in regex, and how do you create them?

28. Explain the purpose of the `re.VERBOSE` flag in `re.compile`.

29. Write a regex pattern to match valid credit card numbers (e.g., Visa or MasterCard).

30. How do you use lookaheads and lookbehinds in regex patterns, and what are they used for?