ASSIGNMENT - 16

Q1. What is the benefit of regular expressions?

Ans: Regular expressions offer a powerful way to search, match, and manipulate text. Some benefits include:

* Flexibility: They allow for complex pattern matching within text data.
* Efficiency: They can perform operations like search and replace more efficiently than manual methods.
* Portability: Regular expressions are supported in various programming languages, making patterns reusable across platforms.

Q2. Describe the difference between the effects of “(ab)c+”; and “a(bc)+.” Which of these, if any, is the unqualified pattern “abc+”?

Ans: Difference between "(ab)c+" and "a(bc)+":

* "(ab)c+": This pattern looks for a sequence where "ab" is followed by one or more occurrences of "c."
* "a(bc)+": This pattern looks for a sequence where "a" is followed by one or more occurrences of "bc."

Neither of these patterns is the unqualified "abc+". The unqualified pattern "abc+" would look for "a" followed by one or more occurrences of "b," followed by one occurrence of "c."

Q3. How much do you need to use the following sentence while using regular expressions?

import re

Ans: This line is necessary when working with regular expressions in Python. It imports the re module, which provides support for working with regular expressions.

Q4. Which characters have special significance in square brackets when expressing a range, and under what circumstances?

Ans: In square brackets, certain characters have special meanings:

* Hyphen (-): Indicates a range when used between characters. For example, [a-z] represents all lowercase letters.
* Caret (^): Negation or exclusion when used as the first character inside square brackets. For example, [^0-9] matches anything that is not a digit.
* Backslash (): Escapes special characters within square brackets. For instance, [\[\]] matches square brackets.

Q5. How does compiling a regular-expression object benefit you?

Ans: Compiling a regular expression into an object using re.compile() provides efficiency when the pattern will be used multiple times. It pre-processes the regular expression, resulting in faster matching operations.

Q6. What are some examples of how to use the match object returned by re.match and re.search?

Ans: Using match objects returned by re.match and re.search:

* re.match: Returns a match object if the pattern is found at the beginning of the string. Use methods like .group() or .span() to retrieve the matched text or its position.
* re.search: Searches for the pattern anywhere in the string and returns a match object for the first occurrence found. Similar methods can be used to retrieve matched text or positions.

Q7. What is the difference between using a vertical bar (|) as an alteration and using square brackets as a character set?

Ans: Difference between vertical bar (|) and square brackets ([]):

* Vertical Bar (|): Denotes alternation, matching either of the patterns separated by it. For example, a|b matches either "a" or "b."
* Square Brackets ([]): Defines a character set, matching any one of the characters within the brackets. For instance, [abc] matches "a," "b," or "c."

Q8. In regular-expression search patterns, why is it necessary to use the raw-string indicator (r)? In replacement strings?

Ans: Necessity of using the raw-string indicator (r):

* In regular expressions: It is useful to use the raw-string indicator (r) to avoid unintended escapes. For instance, r'\n' represents a newline character, whereas '\n' would represent a backslash followed by "n."
* In replacement strings: Using r in replacement strings isn't necessary, as the purpose of raw strings is to handle special characters during pattern matching rather than in replacement strings. However, using raw strings in replacement strings doesn’t cause any issues.]