**Assignment - 17**

Q1. Explain the difference between greedy and non-greedy syntax with visual terms in as few words as possible. What is the bare minimum effort required to transform a greedy pattern into a non-greedy one? What characters or characters can you introduce or change?

Ans: Greedy syntax matches as much text as possible, while non-greedy syntax matches as little as possible. To transform a greedy pattern into a non-greedy one, introduce a question mark after the quantifier. For example, changing \* to \*?.

Q2. When exactly does greedy versus non-greedy make a difference?  What if you're looking for a non-greedy match but the only one available is greedy?

Ans: Greedy versus non-greedy makes a difference when there are multiple matches in the text. If you're looking for a non-greedy match but only a greedy one is available, you might not get the desired result, such as capturing more text than intended.

Q3. In a simple match of a string, which looks only for one match and does not do any replacement, is the use of a nontagged group likely to make any practical difference?

Ans: In a simple match of a string without replacement, using a non-tagged group is unlikely to make a practical difference since it doesn't affect the outcome of the match.

Q4. Describe a scenario in which using a nontagged category would have a significant impact on the program's outcomes.

Ans: Using a non-tagged category can have a significant impact when performing complex pattern matching or replacement tasks, especially when dealing with overlapping patterns or nested structures.

Q5. Unlike a normal regex pattern, a look-ahead condition does not consume the characters it examines. Describe a situation in which this could make a difference in the results of your programme.

Ans: A look-ahead condition that does not consume characters can make a difference when you need to check for a specific pattern without including it in the match result, such as validating passwords without including certain characters.

Q6. In standard expressions, what is the difference between positive look-ahead and negative look-ahead?

Ans: Positive look-ahead asserts that a pattern must be present ahead in the text, while negative look-ahead asserts that a pattern must not be present ahead in the text.

Q7. What is the benefit of referring to groups by name rather than by number in a standard expression?

Ans: Referring to groups by name in a standard expression improves readability and maintainability of the code, especially when dealing with complex patterns or multiple groups.

Q8. Can you identify repeated items within a target string using named groups, as in "The cow jumped over the moon"?

Ans: Yes, you can identify repeated items within a target string using named groups by capturing the repeated item and referring to it later in the expression.

Q9. When parsing a string, what is at least one thing that the Scanner interface does for you that the re.findall feature does not?

Ans: When parsing a string, the Scanner interface handles tokenization and parsing, providing more structured access to the input compared to re.findall, which returns all matches as a list of strings.

Q10. Does a scanner object have to be named scanner?

Ans: No, a scanner object does not have to be named scanner; you can use any valid variable name to refer to a Scanner object.