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 matching in regular expressions captures as much text as possible (maximal match), while non-greedy or lazy matching captures as little text as possible (minimal match). To transform a greedy pattern into a non-greedy one, you can append a question mark (?) after the quantifier.

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: - The difference between greedy and non-greedy matching becomes significant when there's a possibility of matching smaller or larger portions of the input string. If you're looking for a non-greedy match but the only one available is greedy, the match might be larger than expected.

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, the use of a non-tagged group (also known as a non-capturing group) usually doesn't make a practical difference.

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

Ans: - A scenario where using a non-tagged category would have a significant impact on the program's outcomes could be when you want to use a group for logical grouping or alternation, without capturing the content of the group.

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 in regular expressions could make a difference in the results of your program when you want to assert that certain characters follow your match without including them in the match. This is useful when you want to match something followed or not followed by something else.

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

Ans: - In regular expressions, a positive look-ahead matches a pattern only if it's followed by another pattern, while a negative look-ahead matches a pattern only if it's not followed by another pattern.

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 rather than by number in a regular expression makes the code more organized and readable. It's easier to understand what each group is intended to match when they have descriptive names.

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 in regular expressions.

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: - The Scanner interface in some programming languages (like Java) can parse primitive types and strings using regular expressions, breaking its input into tokens using a delimiter pattern. This is something that the re.findall feature does not do

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

Ans: - No, a scanner object does not have to be named “scanner”. You can choose any valid identifier as the name of a scanner object.