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 matches capture the longest possible segment while non-greedy matches capture the shortest. Transforming from greedy to non-greedy often involves adding a ? after quantifiers like \*, +, or {}.

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 vs. non-greedy matters when matching patterns with varying lengths. If only a greedy match is available but a non-greedy one is needed, adjusting the pattern or using additional constraints might be necessary.

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 single match scenario without replacement, using a non-tagged group might not practically alter the outcome significantly.

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

Ans: When distinguishing but not capturing substrings within a larger pattern, using non-tagged categories can notably impact how the program identifies and processes specific parts of the input string.

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: Look-ahead conditions, by not consuming characters, can affect the selection or exclusion of certain matches in complex patterns, especially when ensuring specific contexts without affecting the overall match.

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

Ans: Positive look-ahead asserts a pattern must follow, while negative look-ahead asserts a pattern must not follow, without consuming the characters.

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

Ans: Using names instead of numbers for groups in regular expressions enhances readability and makes the code more maintainable, especially in complex patterns.

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

Ans: Yes, named groups can help identify repeated items. For instance, in the string "The cow jumped over the moon," a regex pattern could identify repeated words using named groups.

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: Scanner offers parsing control with methods like next() and hasNext(), allowing iteration through tokens one at a time, while re.findall retrieves all matches in one go.

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

Ans: No, a Scanner object doesn't have to be named "scanner." You can choose any valid identifier as the object's name, ensuring it follows naming conventions for clarity and readability in your code.