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: The main difference between Greedy and Non Greedy Match syntax is that Greedy Match tries to find as many repetitions of the quantified pattern as possible, while Non Greedy Match tries to find as few repetitions of the quantified pattern as possible.

import re

print(re**.**findall("a\*", "aaaaa")) *#* Greedy Match

print(re.findall("a\*?", "aaaaa")) # Non Greddy Match

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 Match tries to find as many repetitions of a quantified pattern as possible. Non-greedy matching tries to repeat the quantified pattern as little as possible. If only non-greedy matches are available, other regular expression filtering or pattern matching methods can be used to further identify the desired pattern.

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: There won’t be any difference

import re

phone\_num = re.compile(r'\d\d\d\d')

num = phone\_num.search('The number is 12345678901.')

print(f'Phone number found -> {num.group()}') # Non Tagged group

print(f'Phone number found -> {num.group(0)}') # Tagged Group

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

Ans: Decimal numbers are not marked or captured. This is useful when string value delimiters are not used and the value needs to be captured.

import re

text='123.456'

pattern=r'(\d+)(?:.)(\d+)'

regex\_obj=re.compile(pattern)

matched=regex\_obj.search(text)

matched.groups()

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: Lookahead makes a difference when counting the number of multiple lines or multiple sentences in a string. Without this you cannot get the correct number of lines or sentences in the string.

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

Ans: With forward lookahead, you can add the following conditions: Negative lookahead is similar, but backwards. In other words, allow the pattern to match only if something precedes the pattern.

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

Ans: Referencing groups by name rather than number in standard expressions makes the code clearer and easier to understand.

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

Ans:

import re

text = "The cow jumped over the moon"

p = re.compile('The', re.I)

p.findall(text)

['The', 'the']

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: re.findall() module is used to search for all occurrences that match a given pattern. In contrast, re.search() will only return the first occurrence that matches the specified pattern. re.findall() will iterate over all the lines of the file and will return all non-overlapping matches of pattern in a single step.

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

Ans: Yes.