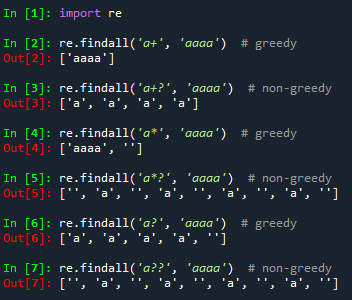
Q1. Explain the difference between greedy and non-greedy syntax with visual terms in as few words as possible. What is the bare minimum effort required to transform a greedy pattern into a non-greedy one? What characters or characters can you introduce or change?

Answer:

The main difference between the two Syntaxes is that the Non Greedy Match will try to match as few as possible, whereas the Greedy Match will try to match as many as possible. To change greedy pattern to non greedy one we need to add a "?" after the repitition operators: "\*?", "+?", “??”, and “{m,n}?”.

Example:



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?

Answer:

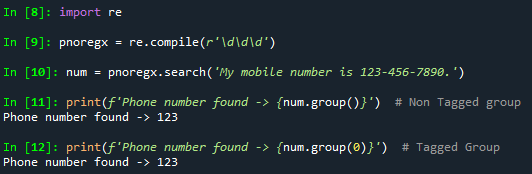
In the non-greedy version of the regex, Python matches the shortest possible string. The longest possible string is matched by Python in the greedy version. We can use other methods of matching if only non greedy match is available.

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?

Answer:

No diffrence will be made in this case.

Example:

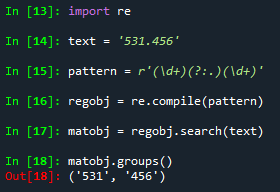


Q4. Describe a scenario in which using a nontagged category would have a significant impact on the program's outcomes.

Answer:

In situations where the separator of value in a string isn't needed, it will be useful. In the below example, the '.' decimal is not tagged or captured.

Example:



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.

Answer:

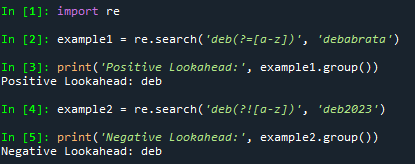
Without the positive look ahead we won’t get the correct count of lines and sentences in a string.

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

Answer:

Positive Look-ahead allows to add a condition for what follows. It looks behind Negative Look ahead. If there is something before the pattern, then it is possible to match it.

Example:



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

Answer:

The benefit of referring to groups by name rather than by number in a standard expression are following: it helps to keep the code clear, easy to understand, and easier to maintain the code.

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

Answer:

import re

text = "The cow jumped over the moon"

reg\_obj = re.compile(r'(?P<w1>The)', re.I)

reg\_obj.findall(text)

Output: ['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?

Answer:

The re.findall() can be used to find occurrences that match a pattern. In contrast, re.search() will be returned the first occurrence that matches the pattern. The 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?

Answer:

The scanner object doesn't need to be named as scanner. It could be any name.