

python_advance_assignment_17

May 31, 2023

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?

```
[ ]: =>The Main difference between Greedy and Non Greedy Match Syntax is The Greedy
      ↳Match will try to match as many
      repetitions of the quantified pattern as possible Whereas The Non Greedy Match
      ↳will try to match as few repetitions
      of the quantified pattern as possible.
```

```
[1]: import re
      print(re.findall("v*", "vvvvvvv")) # Greedy Match Syntax
      print(re.findall("v*?", "vvvvvvv")) # Non Greedy Syntax
```

```
['vvvvvvv', '']
['', 'v', '', 'v', '', 'v', '', 'v', '', 'v', '', 'v', '']
```

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?

```
[ ]: => The Greedy Match will try to match as many repetitions of the quantified
      ↳pattern as possible.
      The Non Greedy Match will try to match as few repetitions of the quantified
      ↳pattern as possible.
      If only Non Greedy Match is available, we can use other filtering or pattern
      ↳matching methods of regex and further
      identify the required 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?

```
[ ]: =>In this Case The Non Tagged Group will not make any difference in this case.
```

```
[3]: import re
      phoneNumRegex = re.compile(r'\d\d\d')
      num = phoneNumRegex.search('My number is 234-567-8901.')
      print(f'Phone number found -> {num.group()}') # Non Tagged group
      print(f'Phone number found -> {num.group(0)}') # Tagged Group
```

Phone number found -> 234
Phone number found -> 234

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

```
[ ]: =>Here in the below Code Snippet . Decimal is not tagged or captured.  
It will be useful in scenarios where the separator of value in a string is of no  
↳ use and we need to capture only the values.
```

```
[4]: import re  
text='135.456'  
pattern=r'(\d+)(?:.)(\d+)'  
regobj=re.compile(pattern)  
matobj=regobj.search(text)  
matobj.groups()
```

```
[4]: ('135', '456')
```

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 ?

```
[ ]: => While counting the number of multiple lines or multiple sentence in a string  
↳ the positive look ahead makes a difference,  
without which we won't get the correct count of lines or sentences in a string.
```

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

```
[ ]: =>Positive Lookahead allows to add a condition for what follows. Negative  
↳ Lookahead is similar,  
but it looks behind. That is, it allows to match a pattern only if there's  
↳ something before it.
```

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

```
[ ]: =>Referring to groups by name rather than by number in a standard expression  
↳ helps to keep the code clear  
and easy to understand.
```

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

```
[5]: import re  
text = "The cow jumped over the moon"  
regobj=re.compile(r'(?P<w1>The)',re.I)  
regobj.findall(text)
```

```
[5]: ['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 ?

```
[ ]: =>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?

```
[ ]: =>Yes, It may have any name.
```