1, # Ans : Greedy version, Python matches the longest possible string

import re

text = "<Robot is the latest addition to the tech items> <Robot is very advanced> <Robot is a machine>"

greedyregobj=re.compile(r'<.\*>')

match=greedyregobj.search(text)

print(match.group())

#the Non-greedy version of the regex, Python matches the shortest possible string

nongreedyregobj=re.compile(r'<Ro.\*?>')

match1=nongreedyregobj.search(text)

match1.group()

2. # Ans : In the non-greedy version of the regex, Python matches the shortest possible string. In the greedy version,

# Python matches the longest possible string. If only non greedy match is available, we can use other filtering

# or pattern matching methods of regex and further identify the required pattern.

3.

import re

phoneNumRegex = re.compile(r'\d\d\d')

mo = phoneNumRegex.search('My number is 415-555-4242.')

print('Phone number found: ' + mo.group()) # non tagged group

print('Phone number found: ' + mo.group(0))

4. # Ans : Non tagged category :

import re

text='135.135'

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

regobj=re.compile(pattern)

matobj=regobj.search(text)

matobj.groups()

# Here the '.' decimal is not tagged or captured.

# It will useful in scenarios where the separator of value in a string is of no use and we need to capture only the

# values.

('135', '135')

5. # Ans : While counting the number of multiple lines or mulytiple sentence in a string the positive look ahead makes a

# difference, without which we wont get the correct count of lines or sentences in a string.

6. # Ans : Positive look ahead is an assertion continuing the search and extending the string e.g.pattern= r'abc(?=[A-Z])''.

# Here after 'abc', ? is extending the search and says that in the remaining string, first identify the next

# charater should be capitalized character between A and Z, but doesnt consume it.

# Example of Positive lookahead

import re

pat=r'abc(?=[A-Z])'

text="abcABCEF"

regobj=re.compile(pat)

matobj=regobj.findall(text)

print("Positive lookahead:",matobj)

# Negative look head is also an assertion to exclude certain patterns e.g. pattern = r'abc(?!abc)', means

# identify a substring containing

# 'abc' which is not followed by another 'abc'

# Example of Negative lookahead

import re

pat1=r'abc(?!abc)'

text1="aeiouabcabc"

regobj1=re.compile(pat1)

matobj1=regobj1.findall(text)

print("Negative look ahead:",matobj1)

Positive lookahead: ['abc']

Negative look ahead: ['abc']

7. # Ans : The benifit of referring to the groups by name is that

# 1)The code is clear

# 2)It is easier to maimtain the code.

8. # Ans :

import re

text = "The cow jumped over the moon"

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

regobj.findall(text)

['The', 'the']

9. # Ans : re.search() method either returns None (if the pattern doesn’t match), or a re.MatchObject that contains

# information about the matching part of the string. This method stops after the first match, so this is best

# suited for testing a regular expression more than extracting data,whereas

# Return all non-overlapping matches of pattern in string, as a list of strings. The string is scanned

# left to right, and matches are returned in the order found.

10. # Ans : The scanner object need not be named scanner. It may have any name.