1. What is the name of the feature responsible for generating Regex objects?

The feature responsible for generating regular expression objects is the re module. This module provides several functions that allow you to create and manipulate regular expressions, including the compile() function which generates a regular expression object. Once you have a regular expression object, you can use it to search for patterns within strings, replace text, and perform other operations.

2. Why do raw strings often appear in Regex objects?

Raw strings are often used in Regex objects because they allow you to write regular expressions without having to escape special characters using backslashes. In Python, backslashes are used as escape characters in regular strings, which means that if you want to include a literal backslash in a regular expression, you need to escape it with another backslash.

3. What is the return value of the search() method?

The search() method of a regular expression object in Python returns a match object if it finds a match for the regular expression within the specified search string, and None if no match is found.

4. From a Match item, how do you get the actual strings that match the pattern?

To get the actual string(s) that match the pattern from a Match object in Python, you can use the group() method.

5. In the regex which created from the r'(\d\d\d)-(\d\d\d-\d\d\d\d)', what does group zero cover? Group 2? Group 1?

In the regular expression r'(\d\d\d)-(\d\d\d-\d\d\d\d)', group zero (i.e., group(0)) covers the entire match, group one (i.e., group(1)) covers the first capturing group (\d\d\d), and group two (i.e., group(2)) covers the second capturing group (\d\d\d-\d\ d\d\d).

6. In standard expression syntax, parentheses and intervals have distinct meanings. How can you tell a regex that you want it to fit real parentheses and periods?

To match literal parentheses and periods in a regular expression, you need to escape them using a backslash (\). In regex syntax, a backslash before a special character, such as (, ), ., etc., indicates that you want to match the literal character rather than its special meaning.

7. The findall() method returns a string list or a list of string tuples. What causes it to return one of the two options?

The findall() method in Python's re module returns a list of all non-overlapping matches of a regular expression pattern in a given string. Whether the method returns a list of strings or a list of tuples depends on whether the regular expression pattern contains capturing groups or not.

8. In standard expressions, what does the | character meaning?

In regular expressions, the | character (vertical bar or pipe symbol) is used to represent the logical OR operation. It allows you to specify multiple alternatives for a pattern, where any one of the alternatives can match the input string.

9. In regular expressions, what does the character stand for?

In regular expressions, the dot (.) character (also known as the period) is a special character that matches any single character except for a newline. It is often used as a wildcard, representing any character in a pattern.

10. In regular expressions, what is the difference between the + and \* characters?

In regular expressions, the + and \* characters are used as quantifiers to specify how many times the preceding character or group should occur in the input string.

11. What is the difference between {4} and {4,5} in regular expression?

The expression {4} means "exactly four occurrences" & The expression {4,5} means "between four and five occurrences".

12. What do you mean by the \d, \w, and \s shorthand character classes denote in regular expressions?

• \d matches any digit character (equivalent to [0-9]).

• \w matches any alphanumeric character (equivalent to [a-zA-Z0-9\_]).

• \s matches any whitespace character (equivalent to [ \t\n\r\f\v]).

13. What do mean by \D, \W, and \S shorthand character classes denote in regular expressions?

• \D matches any non-digit character (equivalent to [^0-9]).

• \W matches any non-alphanumeric character (equivalent to [^a-zA-Z0-9\_]).

• \S matches any non-whitespace character (equivalent to [^ \t\n\r\f\v]).

14. What is the difference between .\*? and .\*?

In regular expressions, .\* matches zero or more of any character, On the other hand, .\*? is a non-greedy or lazy version of

15. What is the syntax for matching both numbers and lowercase letters with a character class?

[0-9a-z]

16. What is the procedure for making a normal expression in regex case insensitive?

To make a regular expression case-insensitive in Python, you can pass the re.IGNORECASE flag to the re.compile() function, which creates a regular expression object.

17. What does the . character normally match? What does it match if re.DOTALL is passed as 2nd argument in re.compile()?

In regular expressions, the . The (dot) character normally matches any character except for a newline (\n) character. However, if the re.DOTALL flag is passed as the second argument to the re.compile() function, then the dot will match any character, including newline characters.

18. If numReg = re.compile(r'\d+'), what will numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') return?

If numRegex = re.compile(r'\d+'), then calling numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') in Python will replace all occurrences of one or more digits with the string 'X'. Therefore, the returned string will be:

19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to do?

Passing re.VERBOSE as the second argument to re.compile() in Python allows you to use verbose mode when defining a regular expression pattern.

20. How would you write a regex that matches a number with comma for every three digits? It must match the given following:

'42'

'1,234'

'6,368,745'

but not the following:

'12,34,567' (which has only two digits between the commas)

'1234' (which lacks commas)

import re

pattern = re.compile(r'^\d{1,3}(,\d{3})\*$')

strings = ['42', '1,234', '6,368,745', '12,34,567', '1234']

for string in string:

match = pattern.match(string)

if match:

print("{string} is a match")

else:

print("{string} is not a match")

21. How would you write a regex that matches the full name of someone whose last name is Watanabe? You can assume that the first name that comes before it will always be a word that begins with a capital letter. The regex must match the following:

'Haruto Watanabe'

'Alice Watanabe'

'RoboCop Watanabe'

but not the following:

'haruto Watanabe' (where the first name is not capitalized)

'Mr. Watanabe' (where the preceding word has a nonletter character)

'Watanabe' (which has no first name)

'Haruto watanabe' (where Watanabe is not capitalized)

import re

name\_regex = re.compile(r'[A-Z][a-z]\*\sWatanabe')

match = name\_regex.search('Haruto Watanabe')

if match:

print('Match found:', matches.group())

else:

print('No match')

22. How would you write a regex that matches a sentence where the first word is either Alice, Bob, or Carol; the second word is either eats, pets, or throws; the third word is apples, cats, or baseballs; and the sentence ends with a period? This regex should be case-insensitive. It must match the following:

'Alice eats apples.'

'Bob pets cats.'

'Carol throws baseballs.'

'Alice throws apples.'

'BOB EATS CATS.'

but not the following:

'RoboCop eats apples.'

'ALICE THROWS FOOTBALLS.'

'Carol eats 7 cats.'

import re

regex = re.compile(r'^(Alice|Bob|Carol)\s+(eats|pets|throws)\s+(apples|cats|baseballs)\.$', re.IGNORECASE)