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

The feature responsible for generating Regex objects is the re.compile() function.

**Why do raw strings often appear in Regex objects?**

Raw strings (prefixed with r) are often used in Regex objects to avoid having to escape backslashes (\). In regular expressions, backslashes are used for special characters, so raw strings make the patterns more readable.

**What is the return value of the search() method?**

The search() method returns a match object if the pattern is found in the string. If no match is found, it returns None.

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

We can use the group() method on the match object to get the actual strings that match the pattern. For example, match.group(0) returns the entire match, and match.group(1) returns the first captured group.

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

Group 0 covers the entire match, e.g., 123-456-7890.

Group 1 covers the first three digits, e.g., 123.

Group 2 covers the last seven digits, e.g., 456-7890.

**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?**

We can escape parentheses and periods with a backslash to match them literally: \(, \), and \..

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

If the regex pattern contains no groups, findall() returns a list of strings.

If the regex pattern contains groups, findall() returns a list of tuples, where each tuple contains the matched groups.

**In standard expressions, what does the | character mean?**

The | character is the OR operator in regular expressions. It matches either the pattern before or the pattern after the |.

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

The . character matches any single character except for a newline (\n).

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

The + character matches one or more occurrences of the preceding element.

The \* character matches zero or more occurrences of the preceding element.

**What is the difference between {4} and {4,5} in regular expressions?**

{4} matches exactly 4 occurrences of the preceding element.

{4,5} matches between 4 and 5 occurrences of the preceding element.

**What do the \d, \w, and \s shorthand character classes signify in regular expressions?**

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

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

\s matches any whitespace character (spaces, tabs, newlines, etc.).

**What do the \D, \W, and \S shorthand character classes signify in regular expressions?**

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

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

\S matches any non-whitespace character.

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

.\* is a greedy match, meaning it will match as much text as possible.

.\*? is a non-greedy match, meaning it will match as little text as possible.

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

We can use the character class [a-z0-9] to match both lowercase letters and numbers.

**What is the procedure for making a normal expression in regex case-insensitive?**

We can make a regex case-insensitive by passing the re.IGNORECASE flag (or re.I for short) as the second argument to re.compile():

re.compile(pattern, re.IGNORECASE)

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

The . character normally matches any character except a newline (\n).

If re.DOTALL is passed, the . character will also match newlines.

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

The sub() method will replace all occurrences of digits with 'X':

'X drummers, X pipers, five rings, X hen'

**What does passing re.VERBOSE as the second argument to re.compile() allow you to do?**

re.VERBOSE allows us to write regular expressions that are more readable by ignoring whitespace and comments within the pattern.

**How would you write a regex that matches a number with commas for every three digits?**

use the following regex:

r'^\d{1,3}(,\d{3})\*$'

This regex matches numbers like 42, 1,234, and 6,368,745 but not 12,34,567 or 1234.

**How would you write a regex that matches the full name of someone whose last name is Watanabe?**

use the following regex:

r'^[A-Z][a-zA-Z]\* Watanabe$'

This regex matches names like Haruto Watanabe, Alice Watanabe, and RoboCop Watanabe but not haruto Watanabe, Mr. Watanabe, Watanabe, or Haruto watanabe.

**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?**

we can use the following case-insensitive regex:

r'^(Alice|Bob|Carol)\s+(eats|pets|throws)\s+(apples|cats|baseballs)\.$'

This regex matches sentences like Alice eats apples., Bob pets cats., and Carol throws baseballs. but not RoboCop eats apples., ALICE THROWS FOOTBALLS., or Carol eats 7 cats..