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

Answer: - The feature responsible for generating Regex objects is the re module in Python.

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

Answer: - Raw strings (prefixed with r) are used in Regex objects to treat backslashes as literal characters. This is important because backslashes are commonly used in regular expressions to escape special characters, and using raw strings helps avoid conflicts between Python's string escape characters and regular expression escape characters.

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

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

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

Answer: - To get the actual strings that match the pattern from a Match object, you can use the group() method with the argument 0 (or no argument). For example, match.group(0) will return the entire matched string.

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?

Answer: - In the given regex r'(\d\d\d)-(\d\d\d-\d\d\d\d)', group 0 covers the entire matched string, group 1 covers the first three digits before the dash, and group 2 covers the remaining seven digits after the dash.

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?

Answer: - To match literal parentheses and periods in a regex, you need to escape them using a backslash. For example, to match a literal left parenthesis (, you would use \(, and to match a literal period ., you would use \..

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

Answer: - To match literal parentheses and periods in a regex, you need to escape them using a backslash. For example, to match a literal left parenthesis (, you would use \(, and to match a literal period ., you would use \..

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

Answer: - In regular expressions, the | character is the pipe symbol and represents the logical OR operation. It allows you to specify multiple alternatives for matching. For example, a|b matches either 'a' or 'b'.

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

Answer: - In regular expressions, the . character is a special metacharacter that matches any character except a newline. It is often used as a wildcard to match any single character.

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

Answer: - In regular expressions, the + character means "one or more occurrences" of the preceding element. It matches one or more repetitions of the preceding pattern. The * character means "zero or more occurrences" of the preceding element. It matches zero or more repetitions of the preceding pattern.

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

Answer: - In regular expressions, {4} specifies exactly four occurrences of the preceding element, whereas {4,5} specifies between four and five occurrences of the preceding element.

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

Answer: - In regular expressions, \d represents any digit (0-9), \w represents any alphanumeric character (a-z, A-Z, 0-9, and underscore), and \s represents any whitespace character (space, tab, newline).

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

Answer: - In regular expressions, \D represents any non-digit character, \W represents any non-alphanumeric character, and \S represents any non-whitespace character.

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

Answer: - The .*? is a non-greedy match that matches as few characters as possible, while .* is a greedy match that matches as many characters as possible. In other words, .*? will match the shortest possible substring, while .* will match the longest possible substring.

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

Answer: - The syntax for matching both numbers and lowercase letters with a character class is [0-9a-z]. It matches any single character that is either a number (0-9) or a lowercase letter (a-z).

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

Answer: - To make a regular expression case insensitive, you can pass the re.IGNORECASE flag as the second argument to the re.compile() function. For example: re.compile(r'pattern', re.IGNORECASE).

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

Answer: - The . character normally matches any character except a newline. If re.DOTALL is passed as the second argument in re.compile(), then the . character will also match a newline.

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

Answer: - numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') will return the string 'X drummers, X pipers, five rings, X hen'. It replaces all occurrences of one or more digits with the letter 'X'.

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

Answer: - Passing re.VERBOSE as the second argument to re.compile() allows you to add whitespace and comments within the regular expression pattern. This can improve the readability and maintainability of complex regular expressions.

20. How would you write a regex that match 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)

Answer: - To match a number with a comma for every three digits, you can use the following regex pattern: r'^\d{1,3}(,\d{3})*$'.

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 one 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)

Answer: - To match the full name of someone whose last name is Watanabe, you can use the following regex pattern: r'[A-Z][a-zA-Z]+\sWatanabe'.

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.'

Answer: - To match 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 either apples, cats, or baseballs; and the sentence ends with a period, you can use the following regex pattern: r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.'. Remember to use the case-insensitive flag (re.IGNORECASE) when compiling the regex pattern.