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

Ans: In many programming languages, the feature responsible for generating regular expression objects is typically called a regular expression compiler or regular expression engine.

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

Ans: Raw strings are often used in regular expressions to avoid having to escape backslashes, which are common in regular expressions. Since backslashes are also used as escape characters in regular Python strings, using raw strings allows you to write regular expressions more cleanly and readably.

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

Ans: The search() method in Python's re module returns a match object if it finds a match for the regular expression pattern in the string, and None if no match is found.

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

Ans: To retrieve the actual strings that match the pattern from a Match object in Python's re module, you can use the group() method. The group() method returns the string matched by the regular expression. Additionally, you can use the groups() method to return a tuple containing all the subgroups of the match.

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?

Ans:

Group 0 always represents the entire match. In this case, it would be the complete string that satisfies the entire regular expression pattern.

Group 1 corresponds to the first set of parentheses, which is (\d\d\d). This captures three consecutive digits.

Group 2 corresponds to the second set of parentheses, which is (\d\d\d-\d\d\d\d). This captures three digits followed by a hyphen and then four more digits.

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?

Ans: In regular expressions, special characters like parentheses and periods have distinct meanings. To match literal parentheses or periods in a string, you need to escape them using a backslash \ to indicate that you are looking for the actual characters themselves rather than their special meaning within the regular expression syntax.

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

Ans: If the regular expression passed to findall() does not contain any capturing groups, it returns a list of strings. Each string in the list represents the full match for the given pattern.

If the regular expression passed to findall() contains one or more capturing groups, it returns a list of tuples. Each tuple in the list represents a match, and each element in the tuple corresponds to a capturing group in the regular expression.

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

Ans: In regular expressions, the | character, known as the pipe or vertical bar, represents an OR operator. It is used to denote a choice between two or more expressions. The | operator matches either the expression preceding it or the expression following it.

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

Ans: In regular expressions, the | character, known as the pipe or vertical bar, represents an OR operator. It is used to denote a choice between two or more expressions. The | operator matches either the expression preceding it or the expression following it.

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

Ans:

* \* (asterisk): Matches zero or more occurrences of the preceding character.
* + (plus): Matches one or more occurrences of the preceding character.

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

Ans:

* {4}: This notation specifies that the preceding character or group must occur exactly 4 times. It matches only when the specified number of occurrences is met.
* {4,5}: This notation specifies that the preceding character or group must occur at least 4 times and at most 5 times. It matches when the specified range of occurrences is met.

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

Ans:

* \d: This shorthand represents any digit from 0 to 9. It is equivalent to the character range [0-9].
* \w: This shorthand represents any word character, which includes letters (both uppercase and lowercase), digits, and underscores. It is equivalent to the character range [a-zA-Z0-9\_].
* \s: This shorthand represents any whitespace character, including spaces, tabs, and newline characters.

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

Ans:

1. \D: This shorthand represents any character that is not a digit. It is equivalent to the character range [^0-9].
2. \W: This shorthand represents any character that is not a word character. It is equivalent to the character range [^a-zA-Z0-9\_].
3. \S: This shorthand represents any character that is not a whitespace character.

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

Ans:

* .\*? (non-greedy or lazy quantifier): This expression matches any character (except for a newline) 0 or more times, but as few times as possible. It tries to find the shortest possible match.
* .\* (greedy quantifier): This expression matches any character (except for a newline) 0 or more times, but as many times as possible. It tries to find the longest possible match.

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

Ans: [0-9a-z]

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

Ans: To make a regular expression case-insensitive in Python, you can use the re.IGNORECASE flag or the re.I flag as an argument to the various re module functions such as re.match(), re.search(), or re.findall(). This flag enables case-insensitive matching for the regular expression pattern.

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

Ans: In a typical regular expression, the . (dot) character matches any character except a newline character. This means it can match any single character, including letters, digits, special characters, and whitespace, except for newline characters.

When the re.DOTALL (or re.S) flag is passed as the second argument in the re.compile() function, it changes the behavior of the dot (.) character. With the re.DOTALL flag, the dot (.) character 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?

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

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

Ans: Passing re.VERBOSE as the second argument to the re.compile() function in Python allows you to create more readable and organized regular expressions by enabling the use of whitespace and comments within the pattern. This can make complex regular expressions easier to understand and maintain.

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)

Ans: pattern = re.compile(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)

Ans: pattern = re.compile(r'^[A-Z][a-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.'

Ans:

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