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

Ans-The feature responsible for generating Regex objects is the ‘re’ module in Python.

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

Ans-Raw strings are often used in Regex objects because it allows us to avoid having to escape backslashes. Since backslashes are common in regular expressions, using a raw string simplifies the process of creating a regex pattern.

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

Ans-The search() method returns a Match object if the pattern is found in the searched string. If the pattern is not found, it returns ‘None’.

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

Ans-The group() method can be used on the match object to get the actual strings that match the pattern.

**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 covers the entire match. Group 1 covers the first set of parentheses, which in this case is the three-digit area code. Group 2 covers the second set of parentheses, which is the seven-digit phone number.

**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-To match actual parentheses or periods in a regex, we can use a backslash before the character. For example, to match an actual left parenthesis, we would use ( in the regex pattern.

**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-The findall() method returns a list of all non-overlapping matches in the searched string. If the regex pattern contains one or more groups, it returns a list of tuples where each tuple represents one match and contains the groups.

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

Ans-The ‘|’ character in a regex means "or", so it matches either the expression before or after it.

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

Ans-In regular expressions, the . character stands for any character except a newline character.

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

Ans- The + character matches one or more occurrences of the preceding character or group, while the \* character matches zero or more occurrences of the preceding character or group.

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

Ans-{4} matches exactly four occurrences of the preceding character or group, while {4,5} matches between four and five occurrences of the preceding character or group.

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

Ans-In regular expressions, \d represents any digit character (equivalent to [0-9]), \w represents any word character (equivalent to [a-zA-Z0-9\_]), and \s represents any whitespace character (including spaces, tabs, and newlines).

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

Ans-In regular expressions, \D represents any non-digit character (equivalent to [^0-9]), \W represents any non-word character (equivalent to [^a-zA-Z0-9\_]), and \S represents any non-whitespace character.

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

Ans-The .\*? is a non-greedy qualifier that matches as few characters as possible. It will match the shortest possible string. On the other hand, .\* is a greedy qualifier that matches as many characters as possible. It will match the longest possible string.

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

Ans-To match both numbers and lowercase letters with a character class, we can use [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 include the re.IGNORECASE or re.I flag when compiling the regular expression pattern using the re.compile() function. This flag enables case-insensitive matching.

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

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

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

Ans-The numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') will return '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 re.compile() allows us to add comments and whitespace within the regular expression pattern to make it more readable and easier to understand.

**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- The regex that matches a number with commas for every three digits is 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- The regex that matches the full name of someone whose last name is Watanabe is 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- you can use the following regex pattern with case-insensitivity to match the described sentence: ^(Alice|Bob|Carol)\s+(eats|pets|throws)\s+(apples|cats|baseballs)\.$

This pattern matches a sentence that starts with either Alice, Bob, or Carol as the first word, followed by one or more whitespace characters. Then it matches either eats, pets, or throws as the second word, followed by one or more whitespace characters. Finally, it matches either apples, cats, or baseballs as the third word, and ensures that the sentence ends with a period. It will match the specified sentences but not the ones that do not meet the criteria.